COMPUTE!'s VIC-20 COLLECTION

A giant assortment of over 70 never-before-published articles and programs for the VIC-20. Action games, thinking games, utilities, graphics, sound, and tutorials.

A **COMPUTE! Books** Publication

$12.95

COMPUTE!'s VIC-20 COLLECTION

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

Copyright 1985, COMPUTE! Publications, Inc. All rights reserved.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the United States Copyright Act without the permission of the copyright owner is unlawful.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-87455-007-6

The authors and publisher have made every effort in the preparation of this book to insure the accuracy of the programs and information. However, the information and programs in this book are sold without warranty, either express or implied. Neither the authors nor COMPUTE! Publications, Inc. will be liable for any damages caused or alleged to be caused directly, indirectly, incidentally, or consequentially by the programs or information in this book.

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is one of the ABC Publishing Companies and is not associated with any manufacturer of personal computers. VIC-20 is a trademark of Commodore Electronics Limited.

Contents

Foreword vii

Chapter 1. Tutorials

Page Flip / *David P. Albright* 3
Programmable Characters / *Peter Busby* 5
Mixing Text and Graphics Characters / *C. D. Lane* 9
Relative Files / *David Lowrey* 10
Taping Programs / *R. Gordon McOrmond* 13
Control over Input and Output Format / *Keith Schleiffer* 18
VIC Tricks / *John Stilwell* 23

Chapter 2. Utilities

Screen Scroll / *Tim Adcock* 29
Tape Search / *William F. Brydges* 31
Move X / *Bruce Farrington* 34
Speed Sticks / *Charles Gerheim* 36
Arcade-Style Subroutines / *Gary Greenwald* 38
Screen Saves / *Steve Henrickson* 41
Tape to Disk / *John Hollenberg* 44
Super Dump / *Bruce Jordan* 46
Hold a Mirror / *C. D. Lane* 48
Compressed Keyword Lister / *C. D. Lane* 54
Multiprocessing / *C. D. Lane* 57
Screen Print II *and* Big Screen Print / *C. D. Lane* 62
Tape Catalog / *Pierre Pondrom* 64
Microassembler / *Gregory Sommerville* 70
Special Characters in the Expanded VIC / *Royan Webb* 78
Faster BASIC / *Todd Wilson* 81

Chapter 3. Sound and Graphics

Paddle Graphics / *James Calloway* 87
VIC Graph / *Ritchie Falk* 91
Sprite-Imation / *Daniel Gray* 95
VIC Draw McGraw / *Thomas Gronos* 105
Music Mate / *Griffin I. Johnson* 115
Piano Player / *Gene Klco* 120

Bitmapping Pixel / *Paul McManamon* 128
Graph / *Soori Sivakumaran* 130
Rainbow Border / *Doug Smoak* 135
Music Maker / *Brian Zupke* 138

Chapter 4. Applications

Write-On / *F. Wendell Johnson* 153
VIC Marquee / *Louis Mendelsohn* 158
Message Board / *Doug Smoak* 162

Chapter 5. Sports Games

Strike Three / *Tim Adcock* 167
Speed Demon / *Kerry Griffin* 173
Lap Racer / *Jeff Ranney* 178

Chapter 6. Logic and Luck

Slot Machine / *Richard F. Bohr* 185
Code Game / *Linton S. Chastain* 189
Logicolor / *Stephen Hust* 191
Maneuver / *Bryan S. Jones* 195
Battleship / *Bill McDannell* 199
Flipper / *Diana N. Rowland* 203
Poker Challenge / *Harvey J. Stapleton* 205

Chapter 7. Mazes

Heist / *John Clopton and Doug Thorsvik* 211
Diamond Maze / *V. Cortes* 215
Meltdown / *Edward Tekeian* 217
Rats! An Artificial Intelligence Simulation / *Ulrich Merten* 220

Chapter 8. Shoot-em-ups

Space Corridor / *Paul Austin* 235
Base Defense / *Jerry Fennell* 239
Bomber Squadron / *Chris Hill* 241
Ex-Blast / *Marcus Warren Hobbs* 246
Meteors / *Jim Koermer and Bo Short* 249
Hyper Ballshot / *Robert Lewis* 251

Laser Command / *Robert L. Lykins* 254
Space Blockade / *Joe W. Rocke* 258
Torpedo-8 / *Richard Ruef* 261
Sevicog / *Jim Schmitz* 264
Pilot / *Mike Scharland* 266
Laser War / *Rob Westphal and Al Switzer* 270
Challenger One / *Andy Wolber* 273

Chapter 9. Adventures

Space Mission / *Paul F. McMillan* 279
Dungeon Escape / *Richard Woods* 286

Chapter 10. More Games

Place Your Bet / *Bob Appleton* 293
Boogieball / *Bobby Evangelista* 296
Light Cycles / *Robert Flatman* 300
Rescue Mission / *E. L. Hayno* 303
Scavenger Hunt / *Robert L. Lykins* 306
Hide-N-Seek / *Steven R. McCloskey* 312
Chopper Lift / *Steven R. McCloskey* 315
Spring Man / *John P. Murphy* 318
Trench Wars / *Larry Volz* 321

Appendices

A. Beginner's Guide to Typing In Programs 327
B. How to Type In Programs 329
C. The Automatic Proofreader / *Charles Brannon* 332

Index 337
Coupon for ordering disk 339

Foreword

The VIC-20 made its first appearance at a 1981 computer show, and though it was just a prototype then, it was a forecast of things to come. Commodore began delivering VICs in 1981, the same year *COMPUTE!* magazine started supporting the computer. Since then, COMPUTE! Publications has published more than half a dozen books devoted exclusively to the VIC, and continues to cover the machine in both *COMPUTE!* magazine and *COMPUTE!'s Gazette.*

Now, the editors at COMPUTE! Publications have put together their largest collection of never-before-published programs and articles exclusively for the VIC. *COMPUTE!'s VIC-20 Collection* is packed with more than 70 programs and articles, each ready to type in and run on your VIC.

Containing nearly three dozen games, most of which run on the unexpanded VIC, *COMPUTE!'s VIC-20 Collection* will provide hours of fun and entertainment for every member of the family. There are sports games to test your athletic talents, maze and logic games to puzzle over, and of course shoot-em-ups that are just plain fun.

Programmers will find *COMPUTE!'s VIC-20 Collection* full of helpful tips and utilities. From screen dump programs to relative files to programmable characters, even the most experienced programmer will find something useful and new.

There's something here to please every VIC owner. And even if you use only a fraction of the more than 70 programs, you'll be more than satisfied with this book. All the programs are ready to type in and run—and to make program entry virtually error-proof, we've included "The Automatic Proofreader." If you want to save typing time, you can purchase a disk containing all the programs in this book by using the coupon in the back or by calling COMPUTE! Publications toll-free at 1-800-334-0868.

You're sure to find some of the games, utilities, and applications in *COMPUTE!'s VIC-20 Collection* valuable additions to your software library.

Chapter One

Tutorials

David P.
Albright

Page Flip

Like many other VIC owners, I bought the 8K expansion cartridge soon after purchasing my computer. One of the projects I wanted to do involved alternate screening. The owner's manual explores only one screen. In my first attempts I was able to get four screens. Then I discovered four more, giving a total of eight.

There are two memory locations used for screen locations, 36866 and 36869. The value within 36869 aligns the screen on a 1K boundary beginning at 4096. The value of 36866 shifts the screen 512 bytes upward in memory from the 1K boundary of 36869.

When changing screen locations you must let the operating system know where the screen is. This is the purpose of location 648. The value within 648 is the page of memory that the screen resides in. A page is a span of 256 bytes.

The Other VIC Screens

All addresses must be converted to binary. No matter what *legal* screen is used, its address when converted to binary will be a 13-digit binary number. Also, the highest value that can be POKEd is 255. In binary, therefore, the POKEd value can be only an 8-bit number.

Now, here comes the hard part. Refer to page 215, section H, of the *Programmer's Reference Guide.*

Bits 4–6 of 36869 are bits 10–12 of the screen's address.
Bit 7 of 36866 is bit 9 of the screen's address.
Bit 7 of 36869 must be a 1.
Bit 12 of the screen's address must also be a 1 (bit 6 of 36869).

This gives you three bits to play with—bits 5 and 4 of 36869, and bit 7 of 36866. There are eight possible combinations in which to arrange these three bits—eight screens.

Multiple screens have many potential uses. They can be used for animation, varied screen layouts in games, and rapid updating of large, multilayout business reports.

Whenever you use any screen other than the "normal" one, you must move the top of your memory area so that it is above the top of your screen. This is done with location 44. The first location of your storage area must contain a zero. Any changes to the start of BASIC work areas must be done before the program is typed in or read from tape. Massive headaches are the result of not following this rule. You will also use this rule to create your own character sets on the expanded VIC.

"Page Flip" requires at least 8K expansion memory. Before entering or loading Page Flip, enter the following line in direct mode:

POKE 44,34:POKE 8704,0:NEW

Page Flip

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 POKE36869,192:POKE648,16:POKE36866,22                :rem 12
12 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 1":PRINT:PRINT:PRINT"40
   96 TO 4607"                                          :rem 34
15 GET A$:IF A$=""THEN15                               :rem 239
20 POKE36869,192:POKE648,18:POKE36866,150               :rem 65
22 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 2":PRINT:PRINT:PRINT"46
   08 TO 5119"                                          :rem 34
25 GET A$:IF A$=""THEN25                               :rem 241
30 POKE36869,208:POKE648,20:POKE36866,22                 :rem 7
32 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 3":PRINT:PRINT:PRINT"51
   20 TO 5631"                                          :rem 25
35 GET A$:IF A$=""THEN35                               :rem 243
40 POKE36869,208:POKE648,22:POKE36866,150               :rem 60
42 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 4":PRINT:PRINT:PRINT"56
   32 TO 6143"                                          :rem 34
45 GET A$:IF A$=""THEN45                               :rem 245
50 POKE36869,224:POKE648,24:POKE36866,22                :rem 11
52 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 5":PRINT:PRINT:PRINT"61
   44 TO 6555"                                          :rem 42
55 GET A$:IF A$=""THEN55                               :rem 247
60 POKE36869,224:POKE648,26:POKE36866,150               :rem 64
62 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 6":PRINT:PRINT:PRINT"65
   56 TO 7167"                                          :rem 51
65 GET A$:IF A$=""THEN65                               :rem 249
70 POKE36869,240:POKE648,28:POKE36866,22                :rem 15
72 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 7":PRINT:PRINT:PRINT"71
   68 TO 7679"                                          :rem 61
75 GET A$:IF A$=""THEN75                               :rem 251
80 POKE36869,240:POKE648,30:POKE36866,150               :rem 59
82 PRINT"{CLR}{3 RIGHT}{3 DOWN}SCREEN 8":PRINT:PRINT:PRINT"76
   80 TO 8191"                                          :rem 52
85 GET A$:IF A$=""THEN85                               :rem 253
```

Peter
Busby

Programmable Characters

Are you looking for more than 64 redefinable characters for your VIC-20? Or perhaps you have 8K+ memory expansion and, much to your distress, have discovered that the capability for high-resolution graphics seems to have disappeared? Here is a routine that will not only give you 256 programmable characters (and more), but will also load *automatically* into a VIC of any memory configuration.

The VIC changes its memory allocations considerably according to how much memory has been added. It is possible to predict, find, and change these values as necessary to accommodate the needs of a particular user program. In essence, there are two variants of VICs: "Low VIC" is the standard 5K version with no more than a 3K module of RAM added. "High VIC" has memory expansion of 8K or more plugged in. Table 1 shows the details of the pertinent memory pointers involved in reserving space for new character sets.

Table 1. Initial Values in BASIC Pointers

Designation	Register	Low VIC	High VIC
Start of BASIC RAM	44	16 or 4	18
Start of variables	46		
Start of arrays	48		
End of arrays	50		
Start of strings	52	30	64 or 96, etc.
End of BASIC RAM	56		
BASIC screen pointer	648	30	16

One consideration must be noted in table 1. If the 16-bit address for the start of BASIC is obtained (by PRINT PEEK (44)*256 + PEEK (43)), it would seem that one byte has vanished. The operating system has set the first byte to zero and incremented the pointer. If the start of BASIC is changed, care must be taken to set the corresponding initial byte to zero. (Otherwise, entering RUN gives a SYNTAX ERROR.)

BASIC presents little difficulty in changing pointers to accommodate the needs of a program, although the procedure for moving itself around under program control becomes somewhat unwieldy. The routine proposed here will therefore include a preprogram to set up the pointers and registers.

The Video Interface Chip adds some rather severe restrictions. The screen-memory start location is determined by the values of four bits, three in register 36869 and one in register 36866. The latter also determines which of two color screen memories is accessed. (It's possible to alternate screens by changing that bit.) The screen memory is restricted to the internal 5K RAM of the VIC and must start at the beginning of an even-numbered page only. (A page is a block of 256 bytes.)

The character-set start location is pointed to by the first four bits in register 36869. Each increment here accesses another character set four pages long. These locations must also be in the internal RAM or ROM of the VIC. See table 2.

Table 2. Access to Character Sets

Procedure: POKE 36869, PEEK (36869) AND 250 OR X where

X=	Start	of Character Set
0	32768	Normal uppercase
1	33792	Reversed uppercase
2	34816	Normal lowercase
3	35840	Reversed lowercase
12	4096	Programmable RAM at page 16
13	5120	Programmable RAM at page 20
14	6144	Programmable RAM at page 24
15	7168	Programmable RAM at page 28

In order to create 64 programmable characters in the Low VIC, it is necessary to drop the end of BASIC and the start of strings (which work backward through memory from the top) down two pages and change the character set pointer to 15:

POKE 56,28:POKE 52,28:POKE 36869,255

This can be done within the user program before any strings are defined; however, that program will not run on a High VIC.

The screen memory may be in any two-page block shown from page 16 to page 30. Redefinable character sets may start only at memory location 4096, 5120, 6144, or 7168.

Program 1 will accommodate 64, 128, or 256 redefinable characters (with case shifting) in a VIC of any memory configuration. The first program is complete in itself and initializes the VIC for subsequent programs.

Two changes can be made. The variable X in the first line should be set to the number of pages of memory required: 2, 4, 6...14. A value of 8 tells the VIC to use 256 programmable characters. Also, the filename VIC SET DEMO in line 8 should be changed to reflect the name of the next program. Change the 8 in that line to a 1 if you're using tape.

Line 1 X is the number of pages to be reserved in RAM. X must be an even number between 0 and 14, inclusive. The rest is identification and credit.

Line 2 Registers V and V−3 point to screen memory. The error routine traps out-of-range values in X.

Line 3 The keyboard buffer counter is set to 2. For Low VICs the end of BASIC is dropped for the reserved memory; X is inverted to count forward; the Boolean operation results in 1 or 0; the program goes to line 5 or ends if the screen doesn't have to move.

Line 4 For High VICs the start of BASIC is raised for the reserved memory. That initial byte of BASIC RAM is set to 0. Flag F is set if X = 2.

Line 5 The screen moves. All those Boolean operations save some bits in the registers and change others according to the value in X.

Line 6 If the new screen occupies the same RAM as this program, half the screen is cleared for the message.

Lines 7–9 The message is written in the new screen area and the program halts, causing BASIC to load and run the next program.

This procedure for calling the next program effectively performs NEW, with all variables set to 0, but with the pointers left at their new values. Consequently, there is no restriction within the bounds of memory available on the length of the next program.

The beginning of the second program should include the first five lines of Program 2 in order to locate the start of the reserved memory block and to keep the user from attempting to run this program without running the first program.

A Demonstration

Program 2 demonstrates the general format of the second program. Type it in and save it with the name VIC SET DEMO. Then run Program 1, which will automatically load and run Program 2.

Line 100 The identification should reflect the name of the user program and credit all authors.

Line 120 Variables are defined and screen locations found.

Lines 130 to 150 The start of reserved memory is found. If the pointers have not been initialized, the error message is printed and the program halts.

Demonstration lines

Line 151 The character set is displayed by POKEing 256 characters to the screen.

Line 160 The Video Interface Chip is pointed to the new character set.
Line 165 The new character set is defined by copying the uppercase/graphics set at 32768.

Program 1. VIC Set

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 X=8:A$="{CLR}VIC SET{DOWN}"                                  :rem 144
2 PRINTA$:V=36869:IFXANDNOT14THENPRINT"{DOWN}IMPROPER X":END
                                                               :rem 51
3 POKE198,2:IFPEEK(56)=30THENPOKE56,30-X:X=14-X:ON(XAND2)/2GO
  TO5:END                                                      :rem 167
4 POKE44,18+X:POKEPEEK(44)*256,0:F=(X=2)                       :rem 34
5 POKEV,PEEK(V)AND207OR4*(XAND12):POKEV-3,PEEK(V-3)AND127OR64
  *(XAND2):POKE648,16+X                                        :rem 228
6 IFFTHENA$=MID$(A$,2):FORF=0TO263:POKEPEEK(648)*256+F,32:NEX
  T:PRINT"{HOME}";                                             :rem 28
7 POKE 36879,15:PRINT"{CLR}{YEL}":POKE 631,13:POKE 632,13
                                                               :rem 108
8 PRINTA$:PRINT"{BLK}{DOWN}LOAD";CHR$(34);"VIC SET DEMO";CHR$
  (34);",8":PRINT"{5 DOWN}RUN":PRINT"{HOME}"                   :rem 166
9 END                                                          :rem 16
```

Program 2. VIC Set Demo

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

Some lines of this program require keywords to be abbreviated so that they will not exceed the four-screen-line limit. See Appendix B.

```
100 PRINT"{CLR}{BLK}{8 SPACES}VIC SET{7 SPACES}{RVS}
    {4 SPACES}DEMONSTRATION{5 SPACES}"                         :rem 76
120 PAGE=256:VIC=36866:COLOUR=37888+4*(PEEK(VI)AND128):SCREEN
    =PEEK(648):POKE VI+13,222                                  :rem 82
130 CHRSET=PEEK(56):HIGH=SC:IF CH=HI THEN CH=CH+2:HI=32
                                                               :rem 22
140 IF CH>32THEN CH=16                                         :rem 176
150 IF CH=HI THEN PRINT"{3 DOWN}{RVS}LOAD{OFF} AND {RVS}RUN
    {OFF} 'VIC SET' FIRST, PLEASE":END                         :rem 124
151 FORI=0TO15:FORJ=0TO15:POKESC*PA+157+J*22+I,J+I*16:POKECO+
    157+J*22+I,INT(RND(0)*8):NEXT:NEXT                         :rem 237
160 POKE VI+3,PEEK(VI+3)AND 240 OR (8+CH/4)                    :rem 14
165 FOR I=(HI-CH)*PA-1TO0STEP-1:POKE CH*PA+I,PEEK(32768+I):NE
    XT                                                         :rem 229
```

C. D.
Lane

Mixing Text and Graphics Characters

The VIC-20 has two independent character sets, graphics and text. The character set pointer can point to only one set at a time. So you can use only one character set at a time, right? Wrong! You can use redefined characters and copy the needed graphics and text characters into user memory. This method has two problems: It allows use of only a subset of both character sets and eats up 512 bytes of program RAM. There is however a solution.

The VIC chip has a raster beam register. This register represents where the VIC is drawing on the screen at any given moment. The trick to mixing character sets is conceptually simple. First, POKE the character pointer to one set, wait until the scan (raster) reaches a certain point on the screen, POKE a different character set pointer, delay a short time, and then change back to the original character set. This procedure is repeated over and over, and different parts of the screen will have different character sets on them. Run the program to verify this for yourself.

The Challenge

The BASIC program is crude at best. It uses the WAIT statement to synchronize the program and the scan. You won't have a fine adjustment where the screen division takes place. However, with machine language you could control the division to within a line or even a character. You could even perhaps change character sets several times during the scan to create several zones.

A further challenge would be to use this technique to mix redefined characters and ROM characters. The screen memory pointer could also be switched in harmony with the scan to create split-screen effects.

The Experiment

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 REM PROGRAM TO PLACE BOTH ROM CHARACTER SETS ON THE SCREEN
   .                                                  :rem 184
20 A=36868:B=A+1:C=240                                :rem 149
30 PRINT"{CLR}";:FORI=0TO504:PRINT" ";:NEXT           :rem 212
40 PRINT"{HOME}GRAPHIC{HOME}{15 DOWN}TEXT"            :rem 157
50 FORI=0TO127:POKE7724+I,I:POKE8054+I,I:NEXT          :rem 1
60 WAITA,128:POKEB,C                                  :rem 189
70 FORI=0TO1:NEXT                                      :rem 81
80 POKEB,C+2:GOTO60                                   :rem 126
```

David
Lowrey

Relative Files

Disk drives have many advantages over cassette tape—two of them are speed and capacity. But the one feature that really separates disk and cassette is the disk drive's ability to perform random access. A tape is, by nature, a sequential device. To get to the ninth record, you must first read records 1 through 8. The 1541 disk drive has the capability to read any of the 683 blocks on a disk in any order.

Random access on the 1541 can be very complicated and usually requires a large amount of code. It is accomplished by telling the disk which track and sector you want to access. This specifies a particular 255-byte block on the disk. If the records are not that large, you end up wasting space on the disk or you must keep track of multiple records per block. You also must have a method of remembering where each block is located.

There is a better way to do all of this. You can use relative files. The format of the file is very simple. It consists of fixed-length records, from 1 to 254 bytes. Each record is individually accessible by specifying its record number. These record numbers are assigned sequentially from 1 to the end of the file. With relative files you don't have to worry about tracks and sectors and buffer positions. Records can span block boundaries so that little space is wasted.

The four major procedures used with relative files are discussed below.

Creating the File

The file is created via the OPEN command. The format is

OPEN *lf*,8,*ch*,"*name*,L,"+**CHR$(***rl*+1**)**

where

lf = logical file number
ch = disk channel (2–14)
name = filename
rl = record length in bytes (1–254)

Example:

OPEN 1,8,2,"RELATIVE,L,"+CHR$(80)

This will create a relative file named RELATIVE. The directory listing will create a relative file named RELATIVE and show a file type of REL. The records would be 79 bytes long. The last character in the record must be a carriage return, CHR$(13). Any attempt to write more than 80 characters would produce an error.

Selecting a Record

The record number you want to access is sent to the disk via the command channel. The format is

PRINT# *lf*,"P"+CHR$(*dc*+96)+CHR$(*rl*)+CHR$(*rh*)+CHR$(*rp*)

where

lf = logical file number that has been opened to the disk command channel (15)
dc = logical file number for the data channel that has been opened to the relative file
rl, rh = record number in low byte/high byte format
rp = byte position in the record where you want to start reading

An easy way to calculate rl and rh is

RH=INT(*record*/256):RL=*record*−(RH*256)

Example:
Assume file 8 has been opened to the data channel and file 15 has been opened to the command channel.

PRINT#15,"P"+CHR$(8+96)+CHR$(10)+CHR$(0)+CHR$(1)

This will set the disk to access record 10 on the next access to logical file 8.

If you attempt to position the file to a nonexistent record, you'll get a disk error of 50, RECORD NOT PRESENT. This may be ignored if you are adding records to the file, but you shouldn't attempt to read a nonexistent record.

Writing Records

Records are written to a relative file using the PRINT# command:

PRINT #1,"RECORD";

This writes six characters to the current record.

PRINT #1,"RECORD"

This writes seven characters (RECORD) plus a carriage return to the current record.

If the record size specified on the OPEN statement is exceeded on a PRINT# statement, a disk error of 51, OVERFLOW IN RECORD, is returned. The record is truncated to the correct length. Remember that carriage returns are counted.

Reading the File

The records may be read using the GET# or the INPUT# statement. GET# reads a byte at a time. INPUT# can be used only to read records of less than 80 bytes.

In the following examples, assume that this code was executed:

```
10 OPEN 15,8,15: REM OPEN COMMAND CHANNEL
20 OPEN 8,8,8,"TESTFILE,L,"+CHR$(100):REM OPEN RELATIVE FILE
25 PRINT #15, "P"+CHR$(8+96)+CHR$(Low Byte)+CHR$(High Byte)+CHR$(1)
 .
 .
 (write records)
 .
 .
 (position to record)
100 A$="" : REM SET INPUT STRING TO NULL
110 GET#8,C$ : REM GET A BYTE FROM FILE 8
120 FOR A=1 to 100
130 A$ = A$ + C$ : REM ADD CHAR TO STRING
140 NEXT
150 REM END OF RECORD
```

This will set the string variable A$ equal to the contents of the current record. If there is a disk error, the program will stop. After the record has been read, the disk must be positioned for the next record with

```
PRINT #15,"P"+CHR$(8+96)+CHR$(Low Byte)+CHR$(High Byte)+CHR$(1)
```

Odds and Ends

A CLOSE command must be executed when you are finished with the file. The logical file must be closed before closing the command channel.

```
CLOSE 8
```

This closes file 8 and causes the disk drive to update the directory.

When a new record is written, all records before the current one are checked to see if they have been created. If they haven't been created, a dummy record, containing CHR$(255) as the first byte, is written to hold its place. If the first record written is record 100, then records 1–99 will be filled in with dummy entries. This could take some time.

If you try to create more records than the disk will hold, you'll get an error of 52, FILE TOO LARGE. Sometimes the VIC will hang up on this error. The only way to recover is to press RUN/STOP–RESTORE. A general estimate of the number of records that will fit on a disk is

```
NR = INT( ( # of free blocks on disk * 255) / record size ) - 5
```

This is just an estimate and should be verified before use.

R. Gordon
McOrmond

Taping Programs

Let's write a short BASIC program consisting of one line:

10 REM

Now let's save the program to tape:

SAVE "PROG1"

Before any of PROG 1 is written to tape, the VIC will write a program header which later will give it information on how to load the program back from tape. You can look at our program header by typing the following line:

FOR J=0 TO 9:PRINT PEEK(828+J);:NEXT

This displays the first ten bytes from the cassette buffer, and it should look like this:

1 1 16 9 16 80 82 79 71 49

The first number (1) is the LOAD command; the next two numbers (1 and 16) are the starting address; 9 and 16 are the ending address; and the last five bytes are the ASCII values of the program's name. The starting and ending addresses refer to the position of the program in memory when it was saved. It takes two bytes, one low byte and one high byte, to store one memory address. To convert the start of the program to decimal, add low byte to high byte times 256:

PRINT 1 + 16 * 256

This gives 4097, the usual starting point for a BASIC program in an unexpanded VIC. The end depends on the length of the program; in our example low byte and high byte convert to 4105. During a SAVE the VIC gets its information on the position of the program from the memory stored in locations 43 and 44 (the start-of-BASIC pointer) and in 45 and 46 (the start-of-variables pointer). Try this:

PRINT PEEK(43) PEEK(44) PEEK(45) PEEK(46)

Saving a Block of Memory

The numbers should be the same as those in the program header. A little experimenting proves that if you POKE new locations into these bytes you can save memory to tape from almost anywhere in RAM. Let's try an experiment. Type

FOR J=6656 TO 6900: POKE J,1: NEXT

This line fills memory from 6656 to 6900 with 1's. Now let's save this memory. First, we must convert the starting and ending addresses into low byte/high byte format. Try this:

X=6656:HIBYTE=INT(X/256):LOBYTE=X−HIBYTE*256

You should find LOBYTE=0 and HIBYTE=26, so we'll adjust our start of BASIC to these values by POKEing the low byte value into location 43 and the high byte into 44:

POKE 43,0:POKE 44,26

Next, substitute 6901 for X and find its low byte and high byte values and POKE them into locations 45 and 46. Why did we use 6901 instead of the end of memory at 6900? Remember, the VIC uses the start-of-variables pointer for the end of memory to be saved. The variables start one byte past the end of the program; therefore, the VIC adjusts for this extra byte by not saving the last byte of a memory block. This can be annoying for machine language programmers who must remember to add one byte to the end of their program SAVEs.

If we have done everything correctly, the VIC thinks it has a BASIC program starting at 6656 and ending at 6900. Now save it with

SAVE "ONE",1,1

If you're not familiar with the purpose of the numbers following the program name, don't worry about it now because we'll talk more about it later. After the SAVE, reset your BASIC pointers with

POKE 43,1:POKE 44,16:NEW

and hold down the RUN/STOP key and hit the RESTORE key.

Then change the memory from 6656 to 6900:

FOR J=6656 TO 6900:POKE J,2:NEXT

Instead of 1's the memory now contains 2's. Rewind the tape and type

LOAD "ONE"

At the end of the LOAD, a PEEK to any memory location between 6656 and 6900 will show a 1 and confirm that this method works for saving memory to tape.

Loading to a Specific Spot in Memory

There is a similar procedure to load a program to any memory address: POKE the starting address into locations 43 and 44, then type the LOAD instruction as you normally do. One use for this is merging BASIC programs. Suppose we want to merge program PROG 1, which we have on tape, with PROG 2, which we'll write now:

NEW

20 REM
30 REM

Normally, if we loaded PROG 1 at this time, it would erase PROG 2, because PROG 1 writes over PROG 2. We can stop this by raising the start of BASIC to the end of PROG 2.

PRINT "LOW BYTE=" PEEK(45):PRINT "HIGH BYTE=" PEEK(46)

This finds the start of variables at low byte=15 and high byte=16. Next, subtract 2 from the low byte (if the low byte was less than 2, you'd use low byte + 254 and high byte − 1) and type

POKE 43,13:POKE 44,16:LOAD "PROG 1"

When the LOAD is finished, type

POKE 43,1:POKE 44,16:LIST

Presto, your merged program. Notice that line 10 is the last line in the program instead of the first. This method won't sort the line numbers in ascending order; however, imaginative use of the text editor can usually correct the problem. Incidentally, don't try to relocate a machine language program this way unless the first two bytes of the program are 0's.

The LOAD Command

The first number in the program header is called the LOAD command. If it is a 1, the VIC can relocate the program to a different memory location from that given in its header. If it is a 3, the VIC must always load the program to the location specified in the header. You can determine which LOAD command will be put in the header if you write out the entire SAVE structure:

SAVE *"filename", device number, secondary address*

Everything but SAVE is optional, but you must include a filename and device number to declare the secondary address. The filename is just your program's name. The device number for a cassette is always 1. The secondary address determines which LOAD command will be written into the header, and it can take one of four values:

0 = writes a LOAD command of 1 to make a relocatable SAVE.
1 = writes a LOAD command of 3 to make a nonrelocatable SAVE.
2 = writes a LOAD command of 1 to make a relocatable SAVE with an end-of-tape marker.
3 = writes a LOAD command of 3 to make a nonrelocatable SAVE with an end-of-tape marker.

If you don't include a secondary address, the default is 0 (LOAD command of 1).This means that the VIC is normally able to relocate any program to the start of BASIC. This is a necessary feature for the VIC, because the start of BASIC changes position depending on its memory configuration.

The secondary address can also be used with the LOAD command, and

the syntax is identical to that of SAVE. With LOAD, the secondary address has one of two meanings:

0 = relocate the program to the start of BASIC.
1 = send the program to the location given in its header.

However, if the LOAD command is 3 (the SAVE command was given with a secondary address of 1 or 3), it will override any instructions given by the LOAD secondary address. The LOAD secondary address has effect only when the LOAD command is 1. We'll demonstrate this by saving a BASIC program that starts at 6657. First, we must write the program, so type

POKE 43,1:POKE 44,26:POKE 6656,0:NEW

This moves the start of BASIC to 6657. Next, write the program:

10 PRINT "THIS PROGRAM WAS"
20 PRINT "WRITTEN STARTING"
30 PRINT "AT MEMORY 6657"

Now, carefully follow these steps:

1. SAVE "PROG SA=0",1,0
2. SAVE "PROG SA=1",1,1
3. NEW:POKE 43,1:POKE 44,16:NEW
4. Rewind tape to the beginning of "PROG SA=0"

Note in step 1 that we could have omitted the device number and the secondary address, and in step 3, we reset the BASIC pointers so that we are back at 4097. Now, if you experiment with different secondary addresses for the LOAD command, you'll find that PROG SA=1 always loads to memory starting at 6657, while PROG SA=0 loads to this location only if you type

LOAD "PROG SA=0",1,1

If instead the LOAD's secondary address is 0, it will relocate to the start of BASIC. The secondary address of a VERIFY command works in exactly the same manner as it does with the LOAD command. When you try to verify program PROG SA=1, it will always compare the program on tape with memory starting at 6657, while program PROG SA=0 can be verified with memory at that location or at the start of BASIC.

SAVE ROM?

In our short exploration of the SAVE, LOAD, and VERIFY commands, we have found that they have several limitations. One that we have not yet mentioned is the inability of the VIC to save memory to tape from above location 32767. Try

POKE 43,0:POKE 44,192:POKE 45,0:POKE 46,224
SAVE "ROM"

If you decipher the low byte/high byte format, you'll see that we are telling the VIC to save memory from 49152 to 57343. This should take several minutes since we are saving a lot of memory (8K). However, after about 20 seconds the SAVE will terminate. The header has been written to tape, but not any of the memory. This means that to save a machine language program from high memory, you must first transfer it down to memory below 32767. To load it back into high memory, you must reverse this procedure or create a relocatable LOAD file with the correct starting address.

Keith
Schleiffer

Control over Input and Output Format

I have always believed that computers are infallible. I realize that programmers make mistakes, but all the machine does is add up numbers and remember where it has put things. It does these two operations much better than a person can. Since I was constantly making arithmetic errors in my checkbook, I got a computer to catch these mistakes and put me back into a life of precision.

It was quite a shock when my computer told me the checkbook balance was $121.69999.

The explanation is simple: The computer has limited precision on the numbers it can manipulate. If the computer repeats a series of steps, like the subtraction needed to balance the checkbook, it will enter from beyond the limit of precision and make a visible change in the number. This is not an error; it's just an imprecise answer. Although the effect of the inaccuracy is reflected throughout the fraction part of the number, it was introduced at the *least significant digit,* the smallest part of the fraction. To avoid this problem we must control the appearance of the output number, concealing any inaccuracy from the user. In BASIC, this requires the PRINT USING statement. But since the VIC's version of BASIC doesn't offer this option, controlling the format must become an arithmetic operation. We must actually change the value of the number printed out to control the format.

Whole Numbers

One way to handle the problem takes advantage of the fact that the computer is accurate when it operates on whole numbers. When you enter a dollars-and-cents value, multiply it by 100 and carry out all the manipulations in cents; then divide by 100 before output to obtain dollars again. This provides a handy solution to the problem in my checkbook program, but there are several disadvantages. Alterations like this must be premeditated—they're easiest when you're writing a new program and can incorporate them carefully. They're not what people expect to find, and they can cause confusion when the program is modified later by someone who doesn't realize (or doesn't remember) what was done. A far bigger problem with this approach is its lack of versatility. It works well enough when you're dealing with dollars and cents, but if you want outputs that reflect a variety of decimal places, this approach may become confusing to manage.

Changing Its Looks

Still, the thought behind this process gives us a place to start. All we really want to do is control the appearance of the output, not the value of the number the computer uses. To do this, we need only modify the output value, not the value in memory. The technique described here simply rounds off the number at the desired place. This conceals the tiny error in essentially the same manner as the PRINT USING statement would. Let's start with that annoying checkbook balance:

```
10 A=121.69999
```

The easiest way to locate a place to cut off the undesirable portion is at the decimal point, between the whole number and fraction. The first step of the rounding process is to multiply by a number large enough to move everything you want above the decimal. The simplest way to do this is to divide by the smallest number you'll expect to see. To round off to cents (hundredths of dollars), divide by 0.01:

```
20 B=A/0.01
```

Right now, if we cut off, or truncate, the rest of the number, we'll still get 12169 instead of 12170. We must still round off the number. While truncating will cut off between 0.99 and 1.0, rounding cuts off between 0.49 and 0.50. If we add 0.5 to the value, then truncate, we'll achieve this effect.

```
30 B=B+0.5
```

The INT function will cut off the fraction, leaving the part we want.

```
40 B=INT(B)
```

To move the decimal place back where it belongs, multiply by the same value that we divided in the first step:

```
50 B=B*0.01
```

And output the result:

```
60 PRINT B
```

This can all be handled in a single statement, relying on no extra variables:

```
60 PRINT INT(A/0.01+0.5)*0.01
```

Using this technique, we need to change only the output statement and will not affect the value of the number the computer is operating on. This method works in any application. If we calculate compound interest, the results will rarely come out in even dollars and cents, but we can use this approach to print out the results in whole cents. This can also be applied to

different numbers of decimal places. If we want only the whole portion of a number, this statement will round if off:

20 PRINT INT(A+0.5)

Or to the thousandth place:

20 PRINT INT(A/0.001+0.5)*0.001

Or use a variable to make the output more versatile:

20 PRINT INT(A/B+0.5)*B

Aligning Numbers

Another problem is that we cannot control the space a displayed number takes up. Normally, output is lined up with the left margin when it is printed out, taking as many columns to the right as needed. For easy reading, a column of figures should line up with the right margin so that all the decimal places are in the same column. In the checkbook program, if we want the numbers displayed in ten columns, including the dollar sign, the decimal point, and the negative sign, we need to figure out how many spaces to print before the number to be sure that the decimal points line up. To do that we need to figure out how big the number is.

```
500 B$=STR$(INT(A/0.01+0.5)*0.01):BL=LEN(B$):IF BL<4 THEN 520
510 IF MID$(B$,BL-3,1)="." THEN 550
520 K=0:FOR L=1 TO BL:IF MID$(B$,L,1)="." THEN K=L:GOTO 540
530 NEXT L:IF K=0 THEN B$=B$+".00":BL=BL+3:GOTO 550
540 IF K=BL-1 THEN B$=B$+"0":BL=BL+1
550 C=9-BL:IF C<=0 THEN 570
560 FOR L=1 TO C:B$=" "+B$:NEXT L
570 B$="$"+B$
580 PRINT B$
590 RETURN
```

This method also allows us to have zeros fill out the columns after the decimal with trailing zeros.

Lines 510–540 are used to force the extra trailing zeros. The advantage of this string approach is that we can edit the output number with more versatility, for instance, by eliminating the sign and printing negative balances in red, bookkeeping style. Numbers that are too large won't fit into the column we've set up, but they won't bring the program to a halt, either. We'll need extra statements and an extra string variable to assemble the output, and the output will run a little bit slower.

GETing Input

In addition to controlling the output appearance of the program, we want to control inputs as well.

```
10 GOSUB 500:C$=B$:PRINT C$:END
500 B$=""
510 GET A$:IF A$="" THEN 510
520 PRINT A$;
530 IF A$=CHR$(13) THEN RETURN
540 B$=B$+A$
550 GOTO 510
```

This is a big improvement over the INPUT statement and will accept a string containing any characters. This is one of the fundamentals of "user-friendly" inputs. A user-friendly program is not just cute messages; the most important part of the concept is to help the user reduce work and avoid making mistakes. The simplest level is to double-check an input number to verify that it fits the desired range. A more effective technique is to limit the characters that can be input so that only the correct characters (in this case, digits) and only the right number of characters can be entered. For the checkbook program, we would type in the amount of the transaction, then hit the C, D, or B key to indicate a check, deposit, or bank charge. All other characters can be ignored.

```
10 GOSUB 500:C=VAL(B$):PRINT C:END
20 REM BRANCH TO AN ACCOUNTING FUNCTION BASED ON THE VALUE OF
   F
500 B$="":F=0:PRINT "$";
510 GET A$:IF A$="C" AND B$<>"" THEN F=1:RETURN
520 IF A$="D" AND B$<>"" THEN F=2:RETURN
530 IF A$="B" AND B$<>"" THEN F=3:RETURN
540 IF (A$>="0" AND A$<="9") OR A$="." THEN PRINT A$;:B$=B$+A
   $
550 GOTO 510
```

This technique is called *masking*, because it works just as though a mask had been placed over most of the keyboard. Line 540 allows the computer to see only the keys we permit. After return from this routine, we use the VAL function to find the input value, and F tells us which arithmetic routine to follow. There is one more key we need to be able to use, INST/DEL. The problem is that if we just include the delete character in the string, then the

VAL function won't see the complete number even though it is displayed properly on the screen. Adding another line:

```
535 IF A$=CHR$(20) THEN B$=LEFT$(B$,LEN(B$)-1):PRINT A$;
```

will cut off the last character of the string rather than leaving the delete character in the string. By masking out the other characters, we have established control over the input value without having to check the value itself, thus limiting the possibility for mistakes.

One drawback to the GET statement is that the cursor is not activated. This can lead to uncertainty at times.

We have discussed some ways to control the format of output to make it more readable. These two concepts, improved readability and reduced possibility of mistakes during user input, are an important part of user-friendly programs.

John
Stilwell

VIC Tricks

Working on machines like the VIC-20 that have limited amounts of memory is challenging. For me, most of the fun is in seeing how little memory I can make a program fit into. This endeavor can be advantageous when working on programs that need all the room they can get. File handlers are a good example. With that example in mind, here are five techniques that have more than an academic value.

Trick 1

This trick is for use with nested FOR-NEXT loops. The NEXT statements have to be consecutive. None of them other than the first one can be branched to. If these requirements are met, the NEXT statements can be combined.

10 FORI=1TO10:FORJ=1TO10:FORX=1TO10:PRINTI;J;X:NEXTX:NEXTJ:NEXTI
memory usage = 65 bytes

This is how the above line looks with the NEXTs combined into one statement:

10 FORI=1TO10:FORJ=1TO10:FORX=1TO10:PRINTI;J;X:NEXTX,J,I
memory usage = 63 bytes

You save one byte for every NEXT statement that you combine.

Trick 2

Use as few variables as possible. They can eat up memory very easily. You can't tell how much your program will take merely by doing the FRE command when the program is loaded into the computer. This is because the computer doesn't allocate memory for variables until it has to use them. Thus, your program will increase in size as it runs. To check the actual size of your program, use the FRE command after the program finishes running.

The best way to use this technique is through organization. One way to cut down on the number of variables that you use is to set aside three variables to be used only for such temporary information as FOR-NEXT loops.

Trick 3

This is something that you shouldn't do until the program is finished. If you aren't familiar with this technique, it can be quite confusing to use while you're writing the program.

Words—or preferably phrases—that are used more than once can be put into string variables. If you use this trick, you obviously can't use a CLR statement in the program without reinitializing the strings.

```
10 PRINT"WE WILL GO TO THE PARK TODAY":GOTO50
20 PRINT"I WILL GO TO THE PARK TODAY":GOTO50
memory usage = 79 bytes
```

In the beginning of the program, set the variable P$ equal to the phrase "WILL GO TO THE PARK TODAY" and then replace this phrase with P$ everywhere it occurs.

```
 5 P$="WILL GO TO THE PARK TODAY"
10 PRINT"WE"P$:GOTO50
20 PRINT"I"P$:GOTO50
memory usage = 74 bytes
```

The more times that you can use P$, the better. You will save a considerable amount of memory each time you use it. In this case, you'll save about 24 bytes each time after the first 2 bytes. The total amount of memory saved is dependent on the length of the string and the number of times that you use it.

Trick 4

This is a technique that I picked up recently. It has the interesting quality of being next to impossible to use. It involves breaking a PRINT statement in half. Normally, if you had a PRINT statement that was longer than 87 characters, you would break it into two separate statements with the first ending with a semicolon.

If there is a TAB or a SPC statement in the PRINT, break it there, thus eliminating the need for a semicolon. The first example shows one of the traditional ways of breaking a PRINT into two separate lines.

```
10 PRINT"HERE I STAND"TAB(9)"AND";
20 PRINT" I AM" Y "YEARS OLD"
```

The next example shows how the first example can be rewritten so that the semicolon is unnecessary.

```
10 PRINT"HERE I STAND"TAB(9)
20 PRINT"AND I AM" Y " YEARS OLD"
```

If you compare the first example with the second, you will notice that the AND is separated from the I AM in the first example, and thus requires two more quotation marks. The total memory saved is three bytes. You save one for the semicolon and two for the two quotation marks not used.

Trick 5

With this trick, we enter into the realm of useful POKEs. Have you ever wanted to write a program that was too big for your VIC? Like a series of adventure games where a game should be played only after the previous one

is completed? If so, the following POKEs will come in handy. POKE 198,1 then POKE 631,131, and your VIC will both load and run the next program on the tape.

These two POKEs do the same job as the SHIFTed RUN/STOP key. As with the RUN/STOP key, words will be printed on the screen. If you don't want anything printed on the screen, the easiest thing you can do is to change the cursor color to match the screen color. You'll still have words printed on the screen, but you won't be able to see them. You will also have to place the cursor on a line where there is nothing printed to avoid getting a SYNTAX ERROR. With this in mind, try it out in the program before you change the cursor color.

Chapter Two

Utilities

Tim
Adcock

Screen Scroll

These two routines will scroll a screen display left or right. They are written for the unexpanded VIC, but will work with 3K expansion. Because of the speed required to accomplish scrolling, they were written in machine language (ML).

The routines work somewhat like the scrolling that occurs when printing to the screen, except everything scrolls left or right. A major limitation of these routines is that they scroll only about one half of the screen (11 lines). This actually is not a big disadvantage since, in many games, you don't have to scroll the whole screen, and, in fact, you may want some things to stay put (such as a player's score). Also included are simple instructions for modifying the programs so that they scroll whatever "block" of the screen you desire.

To use Programs 1 and 2, just type them in as shown. When they are run, the ML portion will be POKEd into memory (in the cassette buffer). Now you can POKE or PRINT anything on the screen (in program mode) as an initial "setup" and call the routine by SYS828; everything on the first 11 lines of the screen should scroll over one space.

As the programs are written now, everything is in white, so if you PRINT your initial setup, be sure it is done in white. You can use other colors, but in your setup you have to POKE these colors into the color matrix. Also, you'll have to use the same color in each individual row. For example, if you want the top row to be in red, you would POKE locations 38400–38421 with 2, PRINT (or POKE) the first line in red, and then call the routine. Of course, before you call the routine, you could also set up the other rows using the same method.

Here is a very basic use of the programs: In a game, you can have an initial setup routine as described above. Then call the routine with SYS828. After everything has scrolled over one space, POKE the empty spaces (on the edge of the screen) with whatever you want. Repeating this process of calling the routine and then POKEing the empty spaces can result in a nice scrolling display. Append Program 3 to Program 1 or 2 for a demonstration.

Program 1. Scroll Right

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
9 POKE3,11                                                    :rem 41
10 FORT=828TO828+60                                           :rem 81
20 READA:POKET,A:NEXT                                         :rem 98
25 DATA 169,20,133,1,169,21,133,2                             :rem 75
27 DATA 166,1,164,2,208,17,162,0,169,32,157,0,30,138,24,105,2
   2,170,201                                                  :rem 93
```

```
30 DATA 220,208,242,96,189,0,30,153,0,30,138,24,105,22,170
                                                    :rem 15
40 DATA 168,200,198,3,165,3,208,237,169,10,133,3,198,1,198,2,
   76,68,3                                          :rem 33
```

Program 2. Scroll Left

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
9 POKE3,11                                          :rem 41
10 FORT=828TO828+62                                 :rem 83
20 READA:POKET,A:NEXT                               :rem 98
25 DATA 169,1,133,1,169,0,133,2                    :rem 231
27 DATA 166,1,164,2,192,21,208,17,162,0,169,32,157,21,30,138,
   24,105,22,170,201                               :rem 231
30 DATA 220,208,242,96,189,0,30,153,0,30,138,24,105,22,170
                                                    :rem 15
40 DATA 168,136,198,3,165,3,208,237,169,11,133,3,230,1,230,2,
   76,68,3                                          :rem 16
```

Program 3. Scroll Demo

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
50 PRINT"{BLK}":PRINT"{CLR}";:                     :rem 217
60 POKE36879,93:FORT=7680+30720TO7899+30720:POKET,0:NEXT
                                                    :rem 41
70 PRINT"{2 SPACES}VIC-20 SCROLLING"                :rem 86
130 PRINT"{3 DOWN} PRESS ANY KEY TO SEE"           :rem 118
170 PRINT"{2 DOWN}{4 SPACES}SCROLL WORK{6 SPACES}"  :rem 157
180 GETA$:IFA$=""THEN180                            :rem 85
190 FORT=1TO30:SYS828:FORG=1TO100:NEXT:NEXT         :rem 32
```

William F. Brydges

Tape Search

"Help! I'll never find that program! It's on one of these tapes in this pile...."

Does this lament sound familiar? There is a way to speed up locating your data files and program files. Can this program help unravel the mystery files?

As more and more programs are stored on tape, that collection of tape cassettes rapidly increases. Some cassettes are used more often than others. Users tend to forget what information is stored on seldom-used cassettes. The question was put to me, "Short of loading every program on every cassette, how can I find out what is on a tape cassette?"

It is a perplexing problem to say the least. Many users know that data files cannot be loaded into memory and executed in the same manner as a program file. This aspect of tape storage presents the problem of identifying the type of file stored on tape as either data or program file. Finally, a program which looks promising to VIC owners may not be able to be loaded into an unexpanded VIC. Is it possible to know the memory required to load a program file before an attempt is made to load it into memory?

Tape Header

Programming the VIC by Raeto Collin West (COMPUTE! Books) contains a lot of useful information about tape storage, including the tape header. When a file is written to tape, an identification header is written that precedes the file information. The header contains information to identify the file as a program or data file, the load address of the file, the end address of the file, and the name of the file, if any. The memory requirement of the program can be calculated from the load and end address by subtraction.

When a tape file is accessed by either LOAD or OPEN, the header of the file is placed into the cassette buffer of the computer. The cassette buffer is at memory location 828 and occupies 192 bytes of memory, of which 191 bytes are usable. To use the identification information stored in the header, the first 5 bytes of the buffer must be PEEKed after the header is loaded from tape. This information is processed as follows:

Byte	Contents
1	Type of file: 1 signifies program file 4 signifies data file
2, 3	Load address of file stored in low byte/high byte order
4, 5	End address of file stored in low byte/high byte order
6–180	Contain name of cassette file, if any (This will contain spaces, chr$(32), if no name present.)

The Program

Line 210 opens logical file 1 to the cassette for a read operation. The information from the file header is placed into the cassette buffer. Lines 220–240 PEEK the value contained in the first 5 bytes of the cassette buffer. These values are assigned to the single array, A. Lines 310 and 320, respectively, calculate the load address and end address of the file. Lines 340–390 PEEK the name of the file one character at a time, until 25 bytes of the name have been determined or until two spaces have been encountered in the name. The name is highlighted when it is printed.

Most BASIC program files will have the load address of 4097. Data files will have their load addresses as the start address of the cassette buffer.

I have used the above routine extensively with my students at school. It has been a valuable utility for searching tape files.

Tape Search

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
180 DIM A(10):COUNT=0                                        :rem 150
195 PRINT"{CLR}":PRINTCHR$(18);"{5 DOWN}THIS ROUTINE WILL LOA
    DAND EXAMINE SUCCESSIVE"                                 :rem 230
200 PRINTCHR$(18);"FILES AS ENCOUNTERED  ON TAPE CASSETTE
                                                             :rem 157
202 PRINTCHR$(18);"{2 DOWN}PRESS ANY KEY                     :rem 185
204 GETR$:IFR$=""THEN 204                                    :rem 113
205 PRINT"{CLR}{5 DOWN}INSERT TAPE CASSETTE  IN DRIVE.
                                                             :rem 200
206 PRINT"TOUCH {RVS}RETURN{OFF} KEY TO   CONTINUE." :rem 143
208 GET C$:IF C$="" THEN 208                                  :rem 91
210 OPEN1,1,0                                                :rem 175
220 FOR I=0 TO 4                                               :rem 8
230 A(I)=PEEK(828+I)                                          :rem 57
240 NEXT I                                                    :rem 30
250 IF A(0)=1 THEN PRINT"{CLR}{2 DOWN}FILE TYPE:  PROGRAM":GO
    TO 300                                                   :rem 197
260 IF A(0)=4 THEN PRINT"{CLR}{2 DOWN}FILE TYPE:   DATA":
                                                             :rem 255
300 REM *** CALCULATE LOAD ADDRESS/END ADDRESS               :rem 181
310 PRINT"{DOWN}LOAD ADDRESS IS:";A(1)+(A(2)*256)            :rem 119
320 PRINT"{DOWN}END ADDRESS IS:";A(3)+(A(4)*256)              :rem 51
330 PRINT"{DOWN}FILE NAME IS:";                              :rem 202
340 FOR I=833 TO 859                                         :rem 235
350 IF PEEK(I)=32 THEN COUNT=COUNT+1                          :rem 98
360 IF COUNT>1 THEN CLOSE1:GOTO390                             :rem 5
370 PRINT CHR$(18);CHR$(PEEK(I));                            :rem 105
380 NEXT I:                                                   :rem 93
390 PRINT                                                     :rem 41
```

```
400 REM *** CALCULATE SIZE OF MEMORY ALLOCATED TO PROGRAM
                                                        :rem 113
410 IF A(0)=4 THEN 450                                   :rem 31
420 PRINT"{DOWN}MEMORY ALLOCATED TO   FILE:";            :rem 18
430 PRINT((A(3)+(A(4)*256))-(A(1)+(A(2)*256)))+1;" BYTES"
                                                         :rem 62
450 PRINT "{4 DOWN}MORE?"                                :rem 32
460 GETJ$:IF J$="" THEN 460                             :rem 105
470 IF J$=CHR$(89) THEN 205                              :rem 94
480 END                                                 :rem 115
```

Bruce
Farrington

Move X

Whether I'm writing an arcade-style game or an educational program, a common subroutine that I find handy is the "Move X" utility program. This program moves a flashing X anywhere on the screen and can check to see if the fire button has been pressed.

This form of input to the computer has several advantages over typing something in by using the keyboard. First, it is quicker and a much more natural way to interact with the computer. Second, small children can learn to use it almost immediately, unlike the typewriter keyboard.

This utility program has several important features: It is easy to change the character to be displayed as well as its color. The flashing X (or whatever character you choose) cannot leave the screen. If you move the X over another character already on the screen, it will restore that character and color back to the screen after the X moves somewhere else. The line numbers are high, so you can append it to the end of an existing program. Another useful feature is that you'll always know your screen character and color code positions. This is very important whether destroying aliens or having children find their way out of a hidden maze.

This utility program is designed to be user friendly. I hope it will be your first step to many new and exciting programs.

Program Description

This program will run on an unexpanded VIC and requires a joystick. Type it in and save it on tape or disk. When you run this program, you will notice a flashing X in the middle of the screen. Using the joystick, practice moving it around the screen. Try to move the flashing X off the screen.

Move X

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
51005 DIM JS(2,2):DD=37154:BI=PEEK(7932):BC=PEEK(38652):PA=37
      137:PB=37152:OB=86:CO=2                            :rem 230
51006 FORI=0TO2:FORJ=0TO2:READJS(J,I):NEXTJ,I             :rem 86
51007 DATA7,0,1,6,8,2,5,4,3                               :rem 75
51009 DEFFNP(S)=(7932+PX+S):DEFFNC(S)=(38652+PX+S)       :rem 117
51020 PX=0:POKE38652,CO:POKE7932,OB                       :rem 55
51030 GOSUB52040:MV=JS(X+1,Y+1)                           :rem 17
51035 IFFNP(0)<=7701AND(MV=0ORMV=1ORMV=7)GOTO51030        :rem 10
51037 IFFNP(0)=7702ANDMV=7GOTO51030                       :rem 29
51040 IFFNP(0)>=8164AND(MV=3ORMV=4ORMV=5)GOTO51030        :rem 16
51045 IFFNP(0)=8163ANDMV=3GOTO51030                       :rem 26
51050 IF(FNP(0)=7680ANDMV=6)OR(FNP(0)=8185ANDMV=2)GOTO51030
                                                         :rem 188
51070 POKEFNP(0),BI:POKEFNC(0),BC:                       :rem 244
```

```
51100 IFMV=0THENPX=PX-22                                  :rem 227
51110 IFMV=4THENPX=PX+22                                  :rem 230
51120 IFMV=6THENPX=PX-1                                   :rem 184
51130 IFMV=2THENPX=PX+1                                   :rem 179
51150 IFMV=7THENPX=PX-23                                  :rem 240
51160 IFMV=1THENPX=PX-21                                  :rem 233
51170 IFMV=5THENPX=PX+21                                  :rem 236
51180 IFMV=3THENPX=PX+23                                  :rem 237
51200 BI=PEEK(FNP(0)):BC=PEEK(FNC(0))                     :rem 101
51400 IFFNP(0)<7680ORFNP(0)>8185THENGOTO51030             :rem 122
51470 POKEFNP(0),OB:POKEFNC(0),CO                         :rem 209
51565 REM THIS IS A GOOD SPOT TO CHECK IF FIRE BUTTON IS DEPR
      ESSED (IF FR=1 THEN...)                             :rem 218
51570 GOTO51030                                           :rem 52
52040 POKEDD,127:S3=-((PEEK(PB)AND128)=0):POKEDD,255      :rem 224
52050 P=PEEK(PA):S1=-((PAND8)=0):S2=((PAND16)=0):S0=((PAND4)=
      0)                                                  :rem 23
52060 FR=-((PAND32)=0):X=S2+S3:Y=S0+S1:RETURN             :rem 183
```

Charles
Gerheim

Speed Sticks

After reading Michael Kleinert's article describing his routine to interface a joystick to the VIC-20 ("Joystick and Keyboard Routine," *COMPUTE!'s Second Book of VIC*), I incorporated the joystick routine into a simple game which worked quite well. The article also got me interested in machine language, and during my reading I found some references concerning the use of the 6522 Versatile Interface Adapters (VIAs), which showed me how to expand the routine to provide for diagonal control by the joystick.

Briefly, each 6522 VIA contains two registers, or ports (A and B), which are used for either input or output. Each port consists of eight bits which contain the input/output information. The data direction (input or output) of each bit is determined by the VIA's Data Direction Register (DDR). To set a bit of a port to input, you must set the corresponding bit of the DDR to 0. For example, to set bit 1 of port A to input and leave the other bits set to output, a BASIC program would POKE 253 (binary 11111101) into the address of the DDR for port A. If a switch is on, the corresponding bit of the port will be 0.

Program 1 is a listing of a routine which loads the joystick routine starting at memory location 7168 ($1C00). The first two statements of the program protect the top 512 bytes of user memory, into which the machine language routine is loaded.

The calling program should perform the following functions:

1. Load the machine language routine.
2. Call the routine with a SYS 7168 statement.
3. Get the direction data from location 7424 ($1D00) with a PEEK(7424).
4. POKE the character into the new location.

Program 2, when added to Program 1, gives a demonstration of how the routine is used. It also gives an example of using the VIAs in a BASIC program. In this case the fire button in the joystick is tested to see if it is being pressed, and the color is changed through a set sequence if it is.

Program 1. Speed Stick

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 POKE52,28:POKE56,28:RP=Ø:DIS=Ø:CO=Ø                    :rem 2Ø8
2 FORI=ØTO6Ø:READN:POKE7168+I,N:NEXT                      :rem 34
9ØØØ DATA169,28        :REM $1CØØ LDA #$1C               :rem 187
9ØØ1 DATA133,2         :REM $1CØ2 STA $Ø2                :rem 95
9ØØ2 DATA169,5Ø        :REM $1CØ4 LDA #$32               :rem 173
9ØØ3 DATA133,1         :REM $1CØ6 STA $Ø1                :rem 99
9ØØ4 DATA169,Ø         :REM $1CØ8 LDA #$ØØ               :rem 121
9ØØ5 DATA168           :REM $1CØA TAY                    :rem 156
9ØØ6 DATA17Ø           :REM $1CØB TAX                    :rem 15Ø
```

```
9007 DATA141,19,145    :REM $1C0C STA $9113          :rem 223
9008 DATA169,127       :REM $1C0F LDA #$7F            :rem 18
9009 DATA141,34,145    :REM $1C11 STA $9122          :rem 205
9010 DATA173,17,145    :REM $1C14 LDA $9111          :rem 181
9011 DATA73,255        :REM $1C17 EOR #$FF           :rem 238
9012 DATA41,28         :REM $1C19 AND #$1C           :rem 143
9013 DATA24            :REM $1C1B CLC                 :rem 72
9014 DATA106,106       :REM $1C1C ROR ROR             :rem 82
9015 DATA24            :REM $1C1E CLC                 :rem 77
9016 DATA14,32,145     :REM $1C1F ASL $9120          :rem 163
9017 DATA176,2         :REM $1C22 BCS #$02           :rem 130
9018 DATA9,8           :REM $1C24 ORA #$08            :rem 54
9019 DATA168           :REM $1C26 TAY                :rem 152
9020 DATA169,255       :REM $1C27 LDA #$FF            :rem 16
9021 DATA141,34,145    :REM $1C29 STA $9122          :rem 208
9022 DATA177,1         :REM $1C2C LDA ($01),Y         :rem 58
9023 DATA141,0,29      :REM $1C2E STA $1D00          :rem 127
9024 DATA96            :REM $1C31 RTS                :rem 107
9025 DATA23,1,45,23,22,0,44,23,24,2,46      :REM $1C32 DATA
                                                     :rem 158
```

Program 2. Speed Stick Demo

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
3 FP=7680:CO=0:CP=38400:FI=160                        :rem 186
10 SYS7168:DIS=PEEK(7424)-23                          :rem 102
20 IFRP+DIS>506THENRP=RP+DIS-506:GOTO40               :rem 209
30 RP=RP+DIS:IFRP<0THENRP=RP+506                       :rem 60
40 POKE7680+RP,102:POKE38400+RP,CO                    :rem 231
50 IF(PEEK(37137)AND32)=0THENCO=CO+1:IFCO>15THENCO=0 :rem 126
60 GOTO10                                               :rem 0
```

Gary
Greenwald

Arcade-Style Subroutines

Have you ever watched an arcade game and wondered how all of the action appears to happen at the same time? For instance, a fire might flicker in a corner, a person walks across the screen, and a bird flies overhead. Anyone who's tried this with BASIC knows that BASIC's too slow—the action appears choppy. Well, with the following machine language subroutines, you too can have sophisticated arcade-style action, and you don't have to understand machine language.

I'll show you two techniques that can be added to your basic game—alternating characters and horizontal motion. Both techniques apply *the interrupt-driven subroutine* which appears to work concurrently with BASIC.

Alternating Characters

This technique allows you to place any character anywhere on the screen and replace that character with a second one of your choice. This repeats indefinitely. You can also control the frequency with which characters alternate. Type in Program 1 and save it before running.

If you typed everything correctly, you will see the letters *A* and *B* alternating very rapidly in the upper-left corner (screen position 7680). Now, let's see how you can make your own alternating characters.

First, select where you want to place your character. Use the following table, where x is your screen location, to determine which values to place in lines 300 and 400.

Desired Screen Location	Column A Line 300	Column B Line 400
7680 to 7935	$x-7680$	30
7936 to 8185	$x-7936$	31

For example, let's say you select the upper-right screen corner, location 7701. In line 300, replace 0 with 21 (7701 − 7680). Line 400 would then remain with a value of 30.

Second, select which characters you want to alternate and place one in location 1 (line 600) and the other in location 2 (line 700) using screen display codes. For instance, if you want to alternate an up arrow with a back arrow, place POKE 1,30 and POKE 2,31 in lines 600 and 700, respectively. (For special effects, use programmable characters. These are explained in your *Programmer's Reference Guide.*)

Third, adjust the frequency of the alternating characters. This is accomplished by POKEing various numbers in locations 37158 and 37159 (lines

900 and 1000). Try 37158,255 and 37159,100. As a rule of thumb, higher values in 37159 will slow the frequency, and vice versa.

A disassembly follows for those who are interested in machine language. This should *not* be typed.

```
LDA  $01
STA  $1E00
LDX  $02
STA  $02
STX  $01
JMP  $EABF
```

Horizontal Motion

This routine allows you to move any character from the left side of the screen to the right, continuously. You pick the character, the original screen location, and the speed. Type in Program 2. In the DATA statements, you will see variables X, Y, and Z. Substitute the following values each time the variables appear: X=0, Y=30, Z=1. Save the program and run.

If you typed in everything correctly, you will see a character (the letter *A)* moving across the screen continuously. Now, let's see how you can program your own horizontal motion.

First, select the initial screen location. Next, go to the previous table and determine the values for columns A and B. Set the variables X and Y in the DATA statements equal to columns A and B, respectively.

Second, set the variable Z equal to your character code.

Third, adjust the speed with various values in locations 37158 and 37159 in lines 600 and 700.

For those of you who are interested, a disassembly follows: Again, *do not* include this in your program.

```
     LDA  $01
     TAX
     LDA  #$20
     STA  $1E00,X
     INX
     CPX  #$15
     BEQ  New
     LDA  #$01
     STA  $1E00,X
     STX  $01
     JMP  $EABF
New  LDA  #$01
     TAX
     STX  $01
     STA  $1E00,X
     JMP  $EABF
```

Some Final Comments

These routines will place a white character in the specified screen location. That's why I have chosen a black screen. If you change the screen color to white, be sure to color the area of character movement.

The machine language is independent of BASIC and will continue *even after BASIC ends.* To stop, hit RUN/STOP–RESTORE at the same time.

Program 1. Alternating

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
100 FORI=828TO841:READA:POKEI,A:NEXT                    :rem 15
200 DATA165,1,141                                      :rem 103
300 DATA0                                              :rem 221
400 DATA30                                              :rem 17
500 DATA166,2,133,2,134,1,76,191,234                    :rem 17
600 POKE1,1                                             :rem 83
700 POKE2,2                                             :rem 86
800 POKE37166,64:POKE788,60:POKE789,3:POKE37166,192:POKE36879
    ,8                                                 :rem 140
900 POKE37158,255                                      :rem 152
1000 POKE37159,255                                     :rem 193
1100 GOTO1100                                          :rem 189
```

Program 2. Moving

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
100 FORI=828TO861:READA:POKEI,A:NEXT                    :rem 17
200 DATA165,1,170,169,32,157,X,Y,232,224,21,240,10,169,Z,157
                                                       :rem 24
300 DATAX,Y,134,1,76,191,234,169,Z                      :rem 34
400 DATA170,134,1,157,X,Y,76,191,234                    :rem 94
500 POKE1,1:POKE37166,64:POKE788,60:POKE789,3:POKE37166,192:P
    OKE36879,8                                         :rem 128
600 POKE37158,255                                      :rem 149
700 POKE37159,255                                      :rem 151
```

Steve
Henrickson

Screen Saves

I had been working on a technique to save a "picture" of the screen on tape for future reference. In my attempts to do so in BASIC, I found it either took an excessive amount of time or far more memory than I could spare.

It wasn't until I read the article by Sheila Thornton in *COMPUTE!'s Second Book of VIC* that I was inspired to use the VIC Kernal routine she discusses in her "Block SAVE and LOAD" article.

Saving the characters that make up a picture is pretty straightforward. This is done on the unexpanded VIC by saving the section of memory containing screen memory (7680–8185). The inability to use the Kernal to save memory above location 32766 ($7FFE) makes it necessary to relocate the color codes located at 38400–38911 to some place that the Kernal can save.

Lines 170–190 contain the loop that relocates the color codes by PEEKing into the old location (OL) and POKEing them into the new location (NL). I chose the two pages (512 bytes) just ahead of screen character codes. This also is the top of available memory. By doing this I was able to save color and characters as one block of memory (7168–8185). Line 120 resets the top of arrays and memory pointers to protect this section of memory.

The 39 bytes of machine code are POKEd into the BASIC input buffer in lines 210–240. The pointers for strings and arrays will be moved when the Kernal routine LOAD is called. Therefore, in lines 360–380 I've stored these pointers just behind the machine code in the input buffer, then reset them after the SYS command in line 390. This brings the total number of bytes required of the input buffer to 43.

In line 250 I've placed the length of the filename in location 0 so that the Kernal routine SETLFS can find it. Because I need to know how much room is necessary for variables and arrays, I've assigned arbitrary values to P and V, which will be assigned new values later. I need to know how much memory is taken for variables and arrays because the filename must be placed where the Kernal routine SETNAM can find it. This is done in lines 260–290. SETNAM can find it by looking at locations 49 and 50 (top-of-arrays pointer).

Lines 450–470 convert the starting and ending addresses to high byte/low byte format and POKE them to available locations at zero page. By directing the SAVE and LOAD routines here, we can control what memory will be saved or loaded.

Once the LOAD routine is called, the memory from 7168–7679 can be transferred back to color memory. This is done with the loop in lines 480–520.

I've intentionally stretched the program out to make it easier to follow. It would save several bytes if it was crunched before merging with another program.

Running the Program

Type in "Screen Saves" and change the 8 in line 540 to a 1 if you're using tape. Save the program before running it. Line 130 allows you to write or draw to the screen using the keyboard. Once you have the desired picture, press the f1 key to save it. Tape users have about ten seconds to press PLAY and RECORD before the computer destroys the picture by printing PRESS RECORD & PLAY ON TAPE. The picture is saved using the filename SCN. The filename can be changed in line 210.

The picture can be loaded by pressing f3. You may now alter the picture, if desired.

Screen Saves

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
120 POKE52,28:POKE56,28                                  :rem 41
130 GETA$:PRINTA$;                                       :rem 64
140 IFPEEK(197)=39THENE=8186:GOTO170                    :rem 119
150 IFPEEK(197)=47THENE=0:A=1:GOTO200                   :rem 179
160 GOTO130                                             :rem 100
170 NL=7168:OL=38400:A=0                                 :rem 62
180 FORM=1TO511                                         :rem 117
190 POKENL,PEEK(OL):OL=OL+1:NL=NL+1:NEXT                 :rem 41
200 RESTORE:A=A+1                                        :rem 11
210 B=7168:R=554:N$="SCN"                                :rem 96
220 FORT=1TO39:READD                                    :rem 230
230 POKER+T,D                                             :rem 5
240 NEXT                                                :rem 213
250 L=LEN(N$):POKE0,L                                   :rem 211
260 P=1:V=1:S=256*PEEK(50)+PEEK(49)+L                    :rem 69
270 FORT=1TOL                                            :rem 49
280 POKES-T,ASC(RIGHT$(N$,T))                           :rem 214
290 NEXT                                                :rem 218
300 SYS555                                               :rem 49
310 V=B:P=252:GOSUB450:IFE=0THEN360                      :rem 95
320 V=E:P=254                                           :rem 207
330 GOSUB450                                            :rem 175
340 SYS574                                               :rem 54
350 GOTO140                                             :rem 102
360 FORT=0TO5                                            :rem 25
370 POKER+T,PEEK(45+T)                                   :rem 36
380 NEXT                                                :rem 218
390 SYS584                                               :rem 60
400 FORT=0TO5                                            :rem 20
```

```
410 POKE45+T,PEEK(R+T)                                  :rem 31
420 NEXT                                               :rem 213
430 IFA=2THEN480                                       :rem 161
440 GOTO140                                            :rem 102
450 POKEP,INT(V/256)                                   :rem 162
460 POKEP-1,V-256*PEEK(P)                              :rem 179
470 RETURN                                             :rem 123
480 OL=7168:NL=38400                                    :rem 90
490 FORM=1TO511                                        :rem 121
500 POKENL,PEEK(OL)                                    :rem 155
510 OL=OL+1:NL=NL+1                                    :rem 108
520 NEXT                                               :rem 214
530 GOTO140                                            :rem 102
540 DATA169,1,162,8,160,0,32,186,255                    :rem 28
550 DATA165,0,166,49,164,50,32,189,255,96               :rem 40
560 DATA169,251,166,253,164,254,32,216,255,96          :rem 239
570 DATA169,0,166,251,164,252,32,213,255,96            :rem 129
```

John
Hollenberg

Tape to Disk

When upgrading from a cassette drive, new owners of disk drives face the daunting task of copying all their tape programs to disk. You could always use your tape drive when you need those programs, but once you've used a disk drive, you quickly get spoiled. It seems unjust to have to go back to slow tape LOADs, but equally unfair to have to copy your entire tape library to disk. What you need is a program to do this for you. After all, automation is one of the strengths of a computer. "Tape to Disk," which requires 32K of memory, can copy an entire tape of programs to disk. All you do is insert a disk with room for the programs, insert the tape, press PLAY, and enter a SYS command. The machine language works for you. You don't need to understand how it works to use it, though.

The program here is a BASIC loader that POKEs the machine language into high memory. The rest of memory is used to load the programs from tape in order to save them to disk. Type in the program and save it—you'll probably want a copy for the future. When you run the program, you'll be asked to wait while the machine language is POKEd in. It will then give you a command to type when you want to start the process: SYS 28672. At this point, insert the tape and rewind it. Insert the disk (be sure there's room for the tape programs), and then enter SYS 28672. You'll be asked to press PLAY on the recorder, and the process begins. The screen shows each tape program being loaded from tape, then saved to disk. You don't have to do a thing. The program doesn't know when to stop, though. After all the programs have been copied to disk, just press RUN/STOP–RESTORE to stop the process. Don't try to exit the program with just RUN/STOP—it won't work. If you get a disk error (flashing red light) during the transfer, you may have to rewind the tape back to the beginning of the program that failed to transfer, then figure out what went wrong with the disk, before you type SYS 28672 again.

Tape to Disk won't copy machine language programs, copy-protected tapes, or any program bigger than 24K. But you'll find it invaluable for copying those tapes full of BASIC programs that everyone seems to accumulate.

Tape to Disk

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}PLEASE WAIT..."                                  :rem 62
20 FORI= 28672 TO 28874 :READDC:POKEI,DC:NEXTI                :rem 140
30 PRINT"{2 DOWN}ENTER {RVS}SYS28672{OFF} TO":PRINT"{DOWN}BEG
   IN TRANSFER."                                               :rem 55
40 DATA 32,146,112,234,234,234,234,234,169,0,162               :rem 48
50 DATA 1,160,0,32,186,255,169,0,32,189,255                    :rem 66
60 DATA 169,0,162,1,160,18,32,213,255,32,161                  :rem 102
```

```
70 DATA 112,169,1,162,8,160,255,32,186,255,169        :rem 226
80 DATA 16,162,65,160,3,32,189,255,173,62,3            :rem 72
90 DATA 201,4,240,79,201,16,240,79,173,61,3            :rem 61
100 DATA 133,251,173,62,3,133,252,174,63,3,172        :rem 202
110 DATA 64,3,169,251,32,216,255,76,8,112,173         :rem 167
120 DATA 61,3,133,251,173,62,3,105,14,133,252         :rem 143
130 DATA 173,64,3,105,14,141,64,3,169,251,76          :rem 113
140 DATA 73,112,173,61,3,133,251,173,62,3,105         :rem 149
150 DATA 2,133,252,173,64,3,105,2,141,64,3            :rem 253
160 DATA 169,251,76,73,112,96,24,76,87,112,24         :rem 183
170 DATA 76,112,112,169,255,133,55,169,90,133,56       :rem 71
180 DATA 32,66,198,32,231,255,96,32,183,255,234        :rem 26
190 DATA 234,201,0,208,1,96,162,0,189,191,112         :rem 156
200 DATA 32,210,255,232,224,10,208,245,76,8,112       :rem 246
210 DATA 234,234,234,13,32,69,82,82,79,82,32          :rem 123
220 DATA 33,13,13,32,32                                :rem 86
```

Bruce
Jordan

Super Dump

"Super Dump" requires the use of the *Super Expander* cartridge and adds a high-resolution screen dump to the menu of *Super Expander* function commands. With Super Dump, the f8 key becomes the Screen Dump key when your VIC is in the hi-res, multicolor, or mixed mode, but retains its function as the LIST key when in the text mode.

After entering the program, be sure to save it to tape or disk before running: The BASIC part of Super Dump clears itself from memory to make room for your programs. After saving, run the program. Within a few seconds, the screen will blink, and the message READY will appear. Now, try entering a graphics program, such as drawing a circle, and then clear the screen. Next, hold down the SHIFT key and hit the f8 key. Your program will list as usual. Notice that Super Dump is gone.

Run your graphics program, and when the image is completed on the screen, hold down the SHIFT key and hit f8 once again. This time, the f8 key signals your VIC to send the image of the screen to the printer, where it's copied.

In your graphics program, if you want to return to the text mode under program control, use the GRAPHIC 0 command instead of the GRAPHIC 4 command. The reason for this is the *Super Expander*'s system architecture: When a program is written with the *Super Expander*, BASIC memory starts at the same location as BASIC memory in an unexpanded VIC, and not in the expansion RAM area. When a GRAPHIC command is executed, the *Super Expander* literally picks up the BASIC program and moves it to the expansion RAM area, then uses the VIC's memory for bitmapping the screen. Reentering the text mode with the GRAPHIC 4 command transfers the program back to the VIC's memory. With the memory pointers changed by Super Dump, the GRAPHIC 4 command has a tendency to mess things up. However, reentering the text mode via the GRAPHIC 0 command leaves your program in the expansion area, where it agrees with the memory pointers. You still have the same amount of memory for your graphics programs it's just in a different location.

Super Dump

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 FORI=3922TO4095:READA:POKEI,A:NEXT                    :rem 66
20 GRAPHIC2:GRAPHIC0:POKE51,81:POKE52,15:POKE55,81:POKE56,15
                                                        :rem 142
30 KEY8,"SYS3922"+CHR$(13):NEW                          :rem 200
40 DATA8,173,200,2,240,7,201,4,240,3                    :rem 254
50 DATA76,98,15,76,237,15,169,0,133,89                  :rem 150
60 DATA133,87,169,16,133,88,169,4,32,177                :rem 248
```

```
70 DATA255,169,0,9,96,32,147,255,169,8          :rem 150
80 DATA32,168,255,162,0,32,185,15,232,224        :rem 15
90 DATA20,240,16,24,165,87,105,160,133,87        :rem 18
100 DATA165,88,105,0,133,88,76,127,15,169        :rem 29
110 DATA13,32,168,255,56,165,87,233,217,133     :rem 123
120 DATA87,165,88,233,11,133,88,230,89,169       :rem 91
130 DATA23,197,89,208,204,32,174,255,32,231     :rem 121
140 DATA255,40,96,169,128,133,90,169,1,133       :rem 77
150 DATA91,169,0,133,92,168,177,87,37,90        :rem 248
160 DATA240,7,24,165,92,101,91,133,92,165        :rem 17
170 DATA91,10,133,91,200,192,7,208,233,24         :rem 7
180 DATA165,92,105,128,32,168,255,165,90,74     :rem 135
190 DATA133,90,208,209,96,169,147,32,210,255    :rem 180
200 DATA162,0,138,157,0,2,232,224,89,208        :rem 214
210 DATA248,32,156,198                          :rem 114
```

C. D.
Lane

Hold a Mirror

We are all familiar with programs that take information in one form and put it out in another. Or programs that calculate data and display it. Here, we will look at two programs which allow your computer to take a look at itself and tell us what it sees. The first program is a simple disassembler, and the second graphically displays the status of the various VIC input devices.

We are constantly entering programs into our computers and running them, usually unaware that what is running our programs is actually another program. Locked into the upper 16K of memory are the BASIC and Kernal ROMs (Read-Only Memories). These are *machine language* programs—programs written in the binary code that digital computers use to process information. These machine language programs may well be larger than any BASIC program we will write for them to run. They control everything we can do and—often more frustrating—everything we cannot do.

We could do more if we knew exactly how the computer works internally. Unfortunately, this understanding is hard to come by. BASIC programs are relatively easy to understand, because BASIC uses English-like instructions such as PRINT and INPUT. But machine language is stored in the computer's memory as numbers which mean little to most people. Before we can make use of the multitude of information stored in the computer, we must have a translator to turn the numbers into symbols that are meaningful to us noncomputers. This is the purpose of the disassembler.

The Disassembler

A *disassembler* is just the opposite of an *assembler*. Few machine language programmers actually program in pure binary numbers any more because the task is so tedious. Instead, they use assemblers, utility programs which allow them to program machine language in three-letter abbreviations called *mnemonics*. The assembler then "assembles" the program—converts it into true machine language. A dissassembler reverses the process, translating the machine language numbers back into three-letter abbreviations we can more easily understand.

Confused? Here's a simple example. Let's say a machine language programmer wants to store the number 15 in memory location 36879 (this will change the VIC's screen color to black and the border color to yellow). In BASIC, we would use the POKE instruction, like this:

POKE 36879,15

When we pressed RETURN, the number would be stored and the screen colors would change. Here is the same thing written in machine language with an assembler:

LDA #$0F
STA $900F

LDA # stands for "LoaD Accumulator with the number immediately following." The accumulator is a temporary place for storing short numbers (0–255) inside the microprocessor chip that runs the computer. The symbol $0F is how the number 15 looks in *hexadecimal,* a base 16 numbering system (don't worry about this now).

STA means "STore Accumulator." It tells the VIC to take the number in the accumulator (the number we just loaded it with) and store it at the following memory location. The symbol $900F is the hexadecimal equivalent of 36879.

The next thing the programmer would do is use the assembler to "assemble" the mnemonics into machine language. Here is what the assembler would produce:

A9 0F
8D 0F 90

This is what machine language looks like in hexadecimal. Converted again to decimal, our familiar base 10 numbering system, the machine language would be stored in the VIC like this:

169, 15, 141, 15, 144

Now you know why machine language subroutines show up in BASIC programs as DATA statements. The numbers in the DATA statements are simply the decimal equivalents of the machine language instructions.

You should now realize exactly what a disassembler does. It just reconverts those decimal numbers back into the mnemonics that we started with.

The disassembler included with this article is particularly suited for the VIC, in that it is small and relatively fast. The program first prompts for an output device to display the mnemonics. The screen, device 3, is the default. Next, it asks for the starting and ending memory addresses in hexadecimal. Some computer books have conversion charts in the back if you need to consult one. The program then proceeds to disassemble instructions between these two points.

Memory locations that are not instructions are output as single bytes of data. Sometimes the disassembler will interpret stray numbers in memory as instructions, too. In your computer you'll find machine language, ASCII strings, keywords, address vectors, program links, and all sorts of information. Consult the table for interesting places to look inside the VIC-20. You can also refer to the memory maps in the back of your manuals. Eventually, it may be your pathway into the world of machine language.

Keytest

This program is a VIC-20 input device diagnostic as well as an excellent demonstration of the VIC's capabilities. Various VIC input devices and their status are displayed about the screen. The upper-left corner of the screen shows the current status of the joystick and its fire button.

The positions of the paddle controllers determine the pitch of the bass and soprano voices. The paddle buttons will move the center dot of the joystick display to the right or left.

The center of the screen contains a display of the VIC keyboard. Normal keys are in white, function keys are in yellow, cursor and control keys are similarly color-coded. The number keys are colored according to the color written on them.

When a key is pressed, it reverses on the screen. You will have to hold down the key and wait a couple of seconds for the screen polling routines to see it. Holding more than one key will cause all of them to reverse. By polling the keyboard ourselves, we are able to detect multiple keys. The VIC decoding routine would have given us only the highest key pressed.

You may notice that when you press more than two keys at once, some extra keys may light up. This is due to the way the keyboard is built. If you have the 8 × 8 table of key rows and columns handy (see "Extended Input Devices," by Mike Bassman and Salomon Lederman, *COMPUTE!'s First Book of VIC*), you will notice that if you press three keys together, the one that completes a "rectangle" on the chart will light.

If a letter key is pressed and the right SHIFT key is depressed, the right-hand graphics character on the key will appear reversed. For example, press A, S, Z, and X simultaneously, and hold the right SHIFT key down. The four card suits will appear. Or hold U, I, J, and K down and press the right SHIFT key; a large circle will appear. This trick does not apply to the left SHIFT key or left graphics character due to their random arrangement on the keys.

At the bottom of the screen you see a circle followed by the word TAPE. The circle is full if a tape cassette key is down, empty otherwise.

The first line of the program disables interrupts, so you may have to hit RUN/STOP–RESTORE several times to get a response. When typing in Program 2, you will want to change the SYS 0 on the first line to SYS 1, until you have it debugged. The program will break if Q, –, STOP, CTRL, 1, 2, space bar, or Commodore keys are pressed while you are using SYS 1, but you will always be able to regain control.

Program 1. Disassembler

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 H$="0123456789ABCDEF":DIMR$(255),T%(3):FORI=3TO0STEP-1:T%(
   I)=16↑I:NEXT                                          :rem 232
20 PRINTCHR$(147);"6502 DISASSEMBLER":PRINT:FORX=0TO255:READR
   $(X):NEXT                                             :rem 129
30 DV=3:INPUT"OUTPUT DEVICE#";DV:PRINT:IFDV<0ORDV>255GOTO30
                                                          :rem 91
40 A$="FFFF":B$="C38A":GOSUB230:OPEN1,DV,1,"ASSEM.OUT":CMD1
                                                         :rem 207
50 FORPC=STOE:S=PEEK(PC):A=PC::GOSUB250:PRINT" ";:A=S:GOSUB25
   0                                                     :rem 162
60 OP$=R$(S):IFOP$=""THENPRINT" .BYTE";:GOTO190          :rem 55
70 PRINT" ";LEFT$(OP$,3);" ";:U=ASC(RIGHT$(OP$,1))-65:rem 142
80 ONUGOTO100,110,120,120,130,140,140,190,150,160,170,180
                                                         :rem 118
90 PRINT"A";:GOTO190                                     :rem 195
100 PRINT"#";:GOTO110                                    :rem 197
110 GOSUB210:GOTO190                                     :rem 178
120 GOSUB210:PRINT",";CHR$(U+85);:GOTO190                :rem 159
130 GOSUB220:GOTO190                                     :rem 181
140 GOSUB220:PRINT",";CHR$(U+82);:GOTO190                :rem 159
150 PC=PC+1:A=PEEK(PC):A=PC+A+1-(AAND128)*2:GOSUB250:GOTO190
                                                          :rem 16
160 GOSUB200:PRINT",X)";:GOTO190                         :rem 169
170 GOSUB200:PRINT"),Y";:GOTO190                         :rem 171
180 PRINT"(";:GOSUB220:PRINT")";                         :rem 138
190 PRINT:NEXT:PRINT:CLOSEDV:END                         :rem 194
200 PRINT"(";                                            :rem 198
210 PC=PC+1:A=PEEK(PC):GOSUB250:RETURN                   :rem 126
220 A=PEEK(PC+2):GOSUB250:GOSUB210:PC=PC+1:RETURN         :rem 41
230 PRINT"ENTER START,END","IN HEXADECIMAL":INPUTB$,A$:GOSUB2
    40:E=S:A$=B$                                         :rem 243
240 S=0:FORI=1TOLEN(A$):B=ASC(MID$(A$,I,1))-48:S=16*S+B+(B>9)
    *7:NEXT:RETURN                                       :rem 170
250 X$="":FORI=3TO0STEP-1:H%=A/T%(I):IFH%=0ANDX$=""ANDI>1GOTO
    270                                                   :rem 18
260 X$=X$+MID$(H$,H%+1,1):A=A-H%*T%(I)                   :rem 213
270 NEXT:PRINTX$;:RETURN                                 :rem 112
300 DATA BRKI,ORAK,,,,ORAC,ASLC,,PHPI,ORAB,ASLA,,,ORAF,ASLF,,
    BPLJ,ORAL,,,,ORAD                                     :rem 38
310 DATA ASLU,,CLCI,ORAY,,,,ORAX                         :rem 167
320 DATA ASLG,,JSRF,ANDK,,,BITC,ANDC,ROLC,,PLPI,ANDB,ROLA,,BI
    TF,ANDF,ROLF,,BMIJ                                   :rem 180
330 DATA ANDC,,,,ANDU,ROLU,,SECI,ANDY,,,,ANDG,ROLG,,RTII,EORK
    ,,,,EORC                                             :rem 208
340 DATA LSRC,,PHAI,EORB,LSRA,,JMPF,EORF,LSRF,,BVCJ,EORL,,,,E
    ORD                                                    :rem 6
350 DATA LSRD,,CLII,EORH,,,,EORG,LSRG,,RTSI,ADCK,,,,ADCC,RORC
    ,,PLAI                                               :rem 111
```

```
360 DATA ADCB,RORA,,JMPM,ADCF,RORF,,BVSJ,ADCL,,,,ADCD,RORD,,S
    EII,ADCH,,,                                          :rem 118
370 DATA ADCG,,,,STAK,,,STYC,STAC,STXC,,DEYI,,TXAI,,STYF,STAF
    ,STXF,,BCCJ                                           :rem 22
380 DATA STAL,,,STYD,STAD,STXE,,TYAI,STAH,TXSI,,          :rem 48
390 DATA STAG,,,LDYB,LDAK,LDXB,,LDYC,LDAC,LDXC,,TAYI,LDAB,TAX
    I,,LDYF,LDAF                                          :rem 71
400 DATA LDXF,,BCSJ,LDAL,,,LDYD,LDAD,LDXE,                :rem 22
410 DATA CLVI,LDAH,TSXI,,LDYG,LDAG,LDXH,,CPYB,CMPK,,,CPYC,CMP
    C,DECC,,INYI,CMPB                                    :rem 183
420 DATA DEXI,,CPYF,CMPF,DECF,,BNEJ,CMPL,,,,CMPD         :rem 151
430 DATA DECD,,CLDI,CMPH,,,,CMPG,DECG,,CPXB,SBCK,,,CPXC,SBCC,
    INCC,,INXI,SBCB                                      :rem 139
440 DATA NOPI,,CPXF,SBCF,INCF,,BEQJ,SBCL,,,,SBCD,INCD,,SEDI,S
    BCH,,,,SBCG,INCG,                                    :rem 248
```

Program 2. Keytest

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

On line 10 of this program, change SYS 0 to SYS 1 until you are certain the program has been entered correctly.

```
10 POKE0,120:POKE1,96:SYS0:POKE56,29:POKE52,29:CLR:V=36864:PO
   KEV+15,10                                             :rem 191
20 DD=37154:SC=7680:SD=V+10:PD=V+8:PA=DD-3:Q=7424:M=255:M1=12
   7:M2=128:QA=SC-1                                      :rem 199
30 M=255:M1=127:M2=128:M3=63:T4=16:T5=32:T6=64:T7=128:TP=8164
                                                         :rem 202
50 DIMJ%(9),K%(63),T%(7):FORI=0TO63:READJ:K%(I)=(ABS(J)+SC+33
   0)*((J<0)*2+1):NEXT                                   :rem 182
60 JY=4:J%(9)=46+SC:FORI=0TO2:FORJ=0TO2:J%(J+I*3)=I*22+J+69+S
   C:NEXTJ,I                                             :rem 159
70 POKEV+14,15:FORI=0TO7:T%(I)=2↑I:NEXT:DEFFNP(X)=INT((M-PEEK
   (X))/2)+M1                                             :rem 77
80 Q1=Q+1:POKEV+14,15:FORI=QTOQ+15:READJ:POKEI,J:NEXT:rem 115
90 PRINT"{CLR}{WHT}"                                    :rem 210
100 PRINT"{RIGHT}U*****I","VIC INPUT"                   :rem 191
110 PRINT"{RIGHT}-{RED}W{WHT}{4 RIGHT}-"                :rem 166
120 PRINT"{RIGHT}-{RIGHT}{GRN} {WHT} {GRN} {WHT}{RIGHT}-","DE
    VICE TEST";                                          :rem 86
130 PRINT"{RIGHT}-{RIGHT} {CYN}Q{WHT} {RIGHT}-"         :rem 235
140 PRINT"{RIGHT}-{RIGHT}{GRN} {WHT} {GRN} {WHT}{RIGHT}-" "
    {8 SPACES}"                                           :rem 1
150 PRINT"{RIGHT}-{5 RIGHT}-"                           :rem 207
160 PRINT"{RIGHT}J*****K","{8 SPACES}"                   :rem 74
170 PRINT"{2 DOWN}{RVS}{22 SPACES}";                    :rem 216
180 PRINT" {YEL}{OFF}COMMODORE{RVS}{K}VIC20{WHT}{4 SPACES}
    {RED}{OFF}Q{RVS}{WHT} ";                             :rem 12
190 PRINT"{44 SPACES}";                                 :rem 166
```

```
195 PRINT"U********************I";                          :rem 73
200 PRINT"- {OFF}←12{RED}3{CYN}4{PUR}5{GRN}6{BLU}7{YEL}8{WHT}
    90+-£{CYN}{2 SPACES}{WHT}{RVS}{G}{OFF}{YEL}F1{RVS}{WHT}-
    ";                                                      :rem 90
210 PRINT"- {OFF}{GRN} {WHT}QWERTYUIOP@*↑{RED}{RVS}{2 SPACES}
    {WHT}{G}{YEL}{OFF}F3{RVS}{WHT}-";                      :rem 151
220 PRINT"-{OFF}{RED} {RVS}{PUR} {WHT}{OFF}ASDFGHJKL:;={CYN}
    {2 SPACES}{WHT}{RVS}O {YEL}{OFF}F5{RVS}{WHT}-";   :rem 212
230 PRINT"-{OFF}{GRN}{2 SPACES}{WHT}ZXCVBNM,./{GRN}{2 SPACES}
    {CYN}{2 SPACES}{WHT}{RVS}{G} {YEL}{OFF}F7{RVS}{WHT}-";
                                                           :rem 254
240 PRINT"-{2 T}P{OFF}{8 SPACES}{RVS}O{4 T}{4 SPACES}-";
                                                           :rem 113
250 PRINT"J********************K";                          :rem 56
260 PRINT"{DOWN}{OFF}{CYN}WTAPE";                          :rem 231
300 FORR=.TO7:POKEQ1,T%(R):SYSQ:N=PEEK(QA):P=R*8:IFR=4THENX=(
    NANDT6)+T7                                              :rem 24
320 FORC=.TO7:Z=K%(P+C):FORG=ABS(Z)TOG-(Z<.):W=PEEK(G)ANDM3
                                                           :rem 132
350 IFNTHENIFT%(C)ANDNTHENPOKEG,W+X:GOTO380                 :rem 26
360 POKEG,W                                                :rem 146
380 NEXT:NEXT:IFR=3THENX=T7                                 :rem 45
390 GOSUB600:NEXT:POKETP,87+6*((JANDT6)=.):GOTO300         :rem 24
600 POKEDD,M1:J=(PEEK(PA)ANDM1)OR(PEEK(PA+1)ANDT7):IFJANDT5TH
    ENPOKEJ%(9),87:GOTO620                                 :rem 128
610 POKEJ%(9),81                                            :rem 84
620 I=4+3*(((JAND4)=.)-((JAND8)=.))-((JANDT4)=T4)+(J>=T7)
                                                            :rem 29
630 IFI<>JYTHENPOKEJ%(I),81:POKEJ%(JY),T5:JY=I             :rem 110
690 POKEDD,M:FORI=.TO1:POKESD+I*2,FNP(PD+I):NEXT:RETURN
                                                            :rem 68
1000 DATA 3,5,7,9,11,13,15,17                              :rem 123
1010 DATA 2,26,28,30,32,34,36,-59                           :rem 71
1020 DATA 24,47,49,51,53,55,57,82                           :rem 93
1030 DATA 45,68,70,72,74,76,78,81                          :rem 105
1040 DATA -95,69,71,73,75,77,-79,-19                       :rem 253
1050 DATA 67,48,50,52,54,56,58,-41                         :rem 140
1060 DATA 25,27,29,31,33,35,37,-63                         :rem 130
1070 DATA 4,6,8,10,12,14,16,-85                            :rem 226
1080 DATA 169,127,73,255,141,32,145,173                    :rem 135
1090 DATA 33,145,73,255,141,255,29,96                       :rem 42
```

C. D.
Lane

Compressed Keyword Lister

Even though your computer's operating system is burned into ROM, the designers have left entrances into it for the purpose of extending it. One such entrance is the CHRGET subroutine in zero page, the often used and discussed *wedge* for adding to BASIC. The *vectors* are another type of entrance into the operating system.

Vectors are addresses written into page 3 of RAM. Various system subroutines execute along and suddenly jump indirectly through these addresses in RAM. These jumps cause execution to continue at the next instruction after the jump; in other words, they appear to do nothing. However, the jumps provide a way out of and back into the computer's ROMs.

Keywords and Tokens

BASIC keywords are stored as single bytes, or tokens, inside the computer. When programs are entered, the keywords are tokenized, or crunched, and when the program is listed, the tokens are translated back into keywords. In the VIC the routine that crunches keywords allows them to be entered in an abbreviated form. For most keywords, typing the first two letters, with the second one SHIFTed, is enough to enter the keyword. Due to similarities in spelling, some keywords use three letters, with the third SHIFTed, as an abbreviation. Other keywords have no abbreviation at all.

Although keywords are entered in abbreviated form, they list in full form, due to the tokenizing procedure. BASIC allows 88 bytes per logical line, but when editing lines we have listed, we are editing expanded keywords. So even though we have filled the line as much as the editor will allow, it can still handle more when crunched. If we fill out a line with abbreviated keywords, we can produce even longer lines that will list correctly, but we will not be able to edit.

One solution to this problem would be to produce listings in the abbreviated keyword format, which we will edit by adding normal or abbreviated keywords and which the crunching routine could read back in. Since we have a link into the listing routine, we can do just that. This solution will not guarantee producing 88-byte lines every time, but it will allow us to come as close as possible and still be able to do screen editing. In fact, in some cases, it will be possible to produce lines longer than the 88-token limit. Caveat programmer!

Abbreviating Keywords

There are three special cases when producing the shortened keywords. Usually, we print the first character as is and SHIFT the second. Two-letter keywords (TO, ON, IF, OR, and so on) do not require abbreviating and are printed normally. Some keywords need two normal characters and a SHIFTed character since they start with the same letters as other keywords (GOSUB, STR$, RETURN, and so on). Finally, there are exceptions which are always printed out in full (INPUT, LOG, NEW, and so on). The first case is handled by a specific test, and the other two are handled by a short table search. One exception which doesn't fall into the above groups is PRINT which prints out as a question mark.

The compressed-listing routine has two user entry points, INIT and KILL. INIT replaces the vector in RAM with the address of the new LIST routine. KILL restores the original vector, returning normal listing. INIT tests the high byte of the vector to make sure that it has not been invoked twice without an intervening KILL.

BASIC keywords are stored in one page (256 bytes) in the beginning of the BASIC ROM. The last letter of each keyword is SHIFTed, marking the end of the keyword. The letters of the keywords are stored in ASCII format, not screen code notation, so a SHIFTed letter has its highest bit turned on. A BMI test detects this. The routine uses a PRINT subroutine to output characters, the only fixed ROM address used besides the keyword table. The routine always returns to the normal LIST routine through the saved vector.

The Loader

The BASIC program provided will load the routine into a VIC-20 with any memory configuration. It partitions off the highest page of memory from BASIC and relocates the machine language routine. The machine code is stored in hexadecimal in the DATA statements of the program. Line 90 is a one-line subroutine that will convert any length hexadecimal string to a decimal value. The locations of the INIT and KILL routines are printed out when the loader is run since they depend on the amount of memory in the computer. Loading takes about ten seconds, and for an unexpanded VIC you should see:

```
RUN
SYS( 7442 ) = INIT
SYS( 7469 ) = KILL
DONE

READY.
```

Doing a subsequent SYS7442 (on an unexpanded VIC) will set the new link and a subsequent LIST will be in compressed format. Even though the listing looks strange, you can still use screen editing, adding normal or compressed keywords. Since the same listing routine is used independently of the output device, you can produce abbreviated listings on the printer or on any other output device. Doing a SYS to the KILL routine restores normal listing, and another call to INIT will turn compressed listing back on. Compressed listing does not affect SAVEs or any other BASIC function.

Compressed List

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 M=PEEK(52)-1:POKE52,M:POKE56,M:POKE51,0:POKE55,0:CLR:M=PEE
   K(52):B=M*256:PC=0                                  :rem 217
20 H$="12":GOSUB90:PRINT"SYS("D+B") = INIT"            :rem 150
30 H$="2D":GOSUB90:PRINT"SYS("D+B") = KILL"            :rem 162
40 READH$:GOSUB90:IFD<>PCTHENPRINT"ERROR IN ";D+100:STOP
                                                         :rem 5
50 FORK=1TO16:READH$:IFH$="END"THENPRINT"DONE":END     :rem 149
60 IFH$="**"THEND=M:GOTO80                              :rem 14
70 GOSUB90                                              :rem 80
80 POKEPC+B,D:PC=PC+1:NEXT:GOTO40                       :rem 80
90 D=0:FORI=1TOLEN(H$):J=ASC(MID$(H$,I,1))-48:D=16*D+J+(J>9)*
   7:NEXT:RETURN                                        :rem 118
100 DATA00,21,0E,49,0F,0D,2A,45,3F,06,36,44,3D,23,3A,10,41
                                                        :rem 156
110 DATA10,00,00,AD,07,03,C9,**,F0,13,8D,AD,**,A9,**,8D,07
                                                        :rem 140
120 DATA20,03,AD,06,03,8D,AC,**,A9,3A,8D,06,03,60,AD,AC,**
                                                        :rem 202
130 DATA30,8D,06,03,AD,AD,**,8D,07,03,60,10,6D,C9,FF,F0,69
                                                        :rem 232
140 DATA40,24,0F,30,65,38,E9,7F,AA,84,49,C9,1A,D0,04,A9,3F
                                                        :rem 247
150 DATA50,D0,53,8D,10,**,A0,FF,CA,F0,08,C8,B9,9E,C0,10,FA
                                                         :rem 16
160 DATA60,30,F5,8C,11,**,A0,08,AD,10,**,D9,07,**,F0,28,88
                                                        :rem 129
170 DATA70,10,F8,A2,00,A0,06,D9,00,**,F0,04,88,10,F8,24,E8
                                                        :rem 159
180 DATA80,AC,11,**,C8,B9,9E,C0,30,1C,20,47,CB,CA,10,F4,C8
                                                        :rem 255
190 DATA90,B9,9E,C0,49,80,D0,10,AC,11,**,C8,B9,9E,C0,30,05
                                                        :rem 220
200 DATAA0,20,47,CB,D0,F5,29,7F,A4,49,A2,00,4C,1A,C7,END
                                                        :rem 201
```

C. D.
Lane

Multiprocessing

This program will allow you to clone your unexpanded VIC into two 1K VICs through software. You will be able to place two separate programs into memory and run them at the same time. The program lets you view them individually or at the same time with a special split-screen display.

Multiprocessing

Multiprocessing means having more than one running program (process) at the same time. Multiprocessing usually takes place on mainframe computers and minicomputers which support many users at the same time, or allow single users to run several programs in parallel. However, even an inexpensive microcomputer can support a multiprocessing environment.

The VIC comes with two processes, the BASIC process you see and the IRQ process which handles the background routines involving the keyboard, clocks, and other system tasks. The two processes do not actually run at the same time; however, they switch back and forth so often that they appear to run in parallel. We'll use the IRQ process to help manage a third process on the VIC, which will look exactly like the normal BASIC process, but will be completely independent of it.

In order to switch processes, we will SYS to the routine PROC which saves the state of the current process on the stack and in reserved blocks of memory. The PROC routine restores a different process from the one it was called by. The next time PROC is called, we pull the switch again and let the original process run, and so on.

To initialize this system, we'll need to have separate memory to store the state of the process that is not running. This requires space to save the first three pages (one page equals 256 bytes) of memory. There is no need to save the fourth page of memory ($0300–$03FF) since it doesn't have any BASIC state information. We will need to save an extra screen memory (two pages) as well as extra color memory for the second BASIC.

We need one page of memory for our program (some of the program we will place in page 3 which is not swapped in the current scheme). This leaves 2K of memory in the unexpanded VIC to split between the two BASIC processes. The memory for the process system lays out as follows:

$9600–$97FF	Color memory
$9400–$95FF	Alternate color memory
$1E00–$1FFF	Screen memory
$1C00–$1DFF	Alternate screen memory
$1900–$1BFF	Storage for the first three pages of dormant process
$1800–$18FF	The process and video control programs
$1400–$17FF	Memory for the second BASIC process

$1000–$13FF Memory for the first BASIC process
$0000–$03FF BASIC and Kernal storage (and process init program)

With careful coding, the system just fits into the unexpanded VIC. By putting the alternate screen memory just below the normal screen memory, we can switch screens by inverting just one bit. This same bit will cause the color memory to switch for us to the location normally used to hold color memory when the VIC is expanded to 8K or more. The initialization part of the process mechanism resides in the tape buffer starting at $033C, and is used only once so the tape buffer can be used normally once processes are running.

Initialization and Process Switches

To initialize the process system, we copy the first three pages of memory into our storage area, copy the screen characters and colors into the alternate screen, and set new BASIC pointers to reflect the locations of the two BASICs. We also need to initialize some of the video registers. Since we copy the state of one process to create the other, both BASIC processes start out as exact copies of each other, but take on their own environments as we program in them.

Process switches are very simple. When we hand off process control, we swap the contents of the first three pages with what we have saved away. Process switches can be done from the keyboard by pressing the f1 key or from a BASIC program with a SYS6275. We also need to update some video information on the process switch if we are not running in *split-screen* mode.

Split Screens

In order to watch both processes at the same time, there is a split-screen routine which runs on the IRQ interrupt. This routine sets the video display to show only 11 rows at a time. During each IRQ interrupt it changes both the screen being displayed and the vertical origin of the screen. One screen starts at the top of the display and the other just below the center so that there is a single-row gap of border color in between.

Normally, when switching screens like this, there is an instant when the screens overlap visually. To avoid this, we wait until the video raster (readable due to the VIC's light pen capabilities) is at the top of the screen before we switch. Although there is only one screen being displayed at a time, they switch so quickly (about 60 times a second) that there appear to be two displays, one above the other.

In order to view or work on the processes separately, press the f3 key, which will switch to single-screen mode. Process switching with the f1 key will also switch to the screen of the process of interest when in single-display mode. Even though only one screen is being displayed, the other

process may still be running, updating its currently invisible screen. Split-screen mode can be turned on again by pressing f3 once more. The same fast switching technique which makes the split screen work can also make the processes appear to run simultaneously.

The process control provided requires that one process let the other run, and vice versa, using the function key or SYS. This is an *explicit handoff* method of control. By having each process constantly handing off to the other, usually in a loop, they will appear to run at the same time, achieving our goal of parallel processing on the VIC. Two example programs which do this are included.

Starting Processes

First, run Program 1, which contains the program in hexadecimal strings and POKEs it into memory. The program contains checksums to catch errors made when typing it in. It takes a little time, so be patient. When it is done, do a SYS828. You should now have a split screen. The next step is the most crucial. Clear the screen by pressing SHIFT–CLR/HOME *several times*. Next do a SYS6275, and press SHIFT–CLR/HOME again *several times*. Repeat this procedure until you are able to switch the cursor easily from window to window by pressing the f1 key.

You are now ready to start using your two BASIC processes, but first, a quick warning: *Do not press RUN/STOP and RESTORE together.* This will terminate the process system, possibly unrecoverably.

To stop a process, simply use RUN/STOP alone. Also be careful when putting processes in infinite loops, since they may be difficult to halt. You can put programs into each BASIC through the normal means—keyboard, tape recorder, disk, or whatever. One of the windows of the split screen will disappear temporarily during tape or disk operations.

The interrupt routine uses a simple test to see if the function keys are down, so they tend to bounce a bit. Quick, short taps are best. You should avoid programs that manipulate storage outside of BASIC or any BASIC pointers, as they will probably crash the program. Programs that are written completely in BASIC (no POKEs) are best. You will have only 1K of memory in each BASIC, so your programs should be short. Look at the example programs to see how to set up simple processes.

Example Programs

Program 2 computes prime numbers, handing off to the process-switch routine before it tests every number. Program 3 displays the VIC character set in large format on the screen, handing off control before each line of display. When testing to see if you have entered them correctly, make sure you put a REM before the SYS statements.

To run the programs, first set up processes as outlined above, load one of the programs, and type RUN. The program should hand off control fairly quickly, and the cursor should appear in the other window. Load the other program and type RUN. Now they both should be running. These processes are best viewed in single-screen mode, and you will be able to verify for yourself that the other process is still running when you are not watching it by seeing how far it has advanced between viewings.

Parallel Processing

We have set up parallel processing on our little VIC in a manner similar to, although much simpler, the way larger computers do. If you examine the machine language program carefully, you will notice that the memory limitations are all that limits it to only one extra BASIC process. Therefore, a VIC with more memory could have even more BASIC processes, each running in turn. This would require special handling of the color memory, among other things, but it is within the VIC's capabilities.

Program 1. Multiprocess

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
100 POKE52,24:POKE56,24:CLR                                  :rem 58
110 FORQ=1000TO1140STEP10                                   :rem 206
120 C=0                                                      :rem 67
130 READADD$,MEM$,CSUM                                       :rem 48
140 H$=ADD$:GOSUB900:ADD=D                                    :rem 2
150 H$=LEFT$(MEM$,2):IFH$=""GOTO210                         :rem 194
160 MEM$=MID$(MEM$,3)                                       :rem 136
170 GOSUB900                                                :rem 177
180 POKEADD,D                                                 :rem 1
190 ADD=ADD+1:C=C+D                                          :rem 49
200 GOTO150                                                  :rem 97
210 IFCSUM<>CTHENPRINT"ERROR IN LINE";Q:END                 :rem 253
220 NEXT                                                    :rem 211
230 PRINT"SYS(828) TO START":PRINT                           :rem 80
240 PRINT"SYS(6275)/F1 TO SWITCH"                           :rem 166
250 PRINT"F3 SPLIT/FULL SCREEN"                             :rem 143
260 POKE52,20:POKE56,20                                      :rem 30
270 NEW                                                     :rem 131
900 D=0:FORI=1TOLEN(H$):J=ASC(MID$(H$,I,1))-48:D=16*D+J+(J>9)
    *7:NEXT:RETURN                                          :rem 166
1000 DATA 033C,78A9188D1503A9008D1403BA8EC318A000B90000990019
     B9,2066                                                 :rem 49
1010 DATA 0354,00019900 1AB90002990 01BB9001E99001CB9001F99001D
     B9,1526                                                  :rem 0
1020 DATA 036C,0096990094B9009799009588D0D3A9008D00148D01148D
     02,2279                                                 :rem 20
```

```
1030 DATA 0384,14A914853885348D2C198D2E198D30198D3219A9188D34
     19,1846                                                  :rem 68
1040 DATA 039C,8D3819A91C8D880285D2A99485F4A94B8DC418A9198DC5
     18,2896                                                 :rem 186
1050 DATA 03B4,A9808DC618A9168D03905860,1323                  :rem 22
1060 DATA 1800,2CC618101BA900CD0490D0FBAC0190ADC4188D01908CC4
     18,2646                                                 :rem 125
1070 DATA 1818,A9804D02908D0290A5C5C927D01B2CC6183010A9074D0F
     90,2386                                                  :rem 88
1080 DATA 1830,8D0F90A9804D02908D0290208318 4C8018C92FD03BA980
     4D,2411                                                  :rem 75
1090 DATA 1848,C6188DC6181009AEC518A016A91B101FA216A91CCD8802
     F0,2394                                                 :rem 136
1100 DATA 1860,02A2968E0290AD01908DC518A219A02EA9022D88020918
     8E,2204                                                  :rem 59
1110 DATA 1878,01908C03908D0F904CBFEA488A489848ADC318BA8EC318
     AA,2848                                                 :rem 199
1120 DATA 1890,9AA000BE0019B9000099001 98A990000BE001AB9000199
     00,1738                                                   :rem 0
1130 DATA 18A8,1A8A990001BE001BB9000299001B8A99000288D0D668A8
     68,2129                                                  :rem 65
1140 DATA 18C0,AA6860,370                                      :rem 4
```

Program 2. Example 1

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 P=1:F=2:M=6275                                              :rem 118
2 P=P+F:J=3:SYSM                                              :rem 250
3 IFJ*J>PTHENPRINTP:GOTO2                                     :rem 191
4 Q=P/J:IFINT(Q)-QTHENJ=J+F:GOTO3                             :rem 118
5 GOTO2                                                       :rem 160
```

Program 3. Example 2

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINTCHR$(147)                                             :rem 220
15 M=32760:P=6275                                             :rem 141
20 S$=CHR$(18):T$=CHR$(146)                                   :rem 173
25 FORI=0TO7:T%(7-I)=2↑I:NEXT                                  :rem 30
30 FORN=0TO511:M=M+8                                          :rem 179
35 PRINTCHR$(19):PRINT:PRINT                                   :rem 63
40 FORJ=MTOJ+7:SYSP                                           :rem 247
45 X=PEEK(J):Y$="  "                                          :rem 246
50 FORK=0TO7                                                  :rem 222
55 IFXANDT%(K)THENY$=Y$+S$                                     :rem 65
60 Y$=Y$+"  "+T$                                              :rem 175
65 NEXT:PRINTY$,Y$                                            :rem 151
70 NEXT:NEXT:GOTO20                                           :rem 244
```

C. D.
Lane

Screen Print II *and* Big Screen Print

"Screen Print" was a program that printed out, bit for bit, whatever was on the VIC screen. Since the original Screen Print program for the VIC, features and optimizations have been incrementally added which make the program both smarter and faster. As a bonus, the "Big Screen Print" variation of Screen Print prints the screen four times larger.

For those unfamiliar with the original Screen Print program, it translated the VIC's 8 × 8 (or 16 × 8) characters into six-bit-high characters for the printer, picking up characters in each row of the screen where it left off, so no bits were missed. A line on the printer could be part of two adjacent lines on the screen. This allowed an *exact* reproduction of the screen, not possible with the printer's 6 × 7 bit character set.

Screen Print II

Certain features which were missing from Screen Print have been incorporated into "Screen Print II." This program is able to print full graphic screens without modification. The height and width of the screen are determined by examining the VIC chip registers rather than being set by the user. The calculation of the number of rows now adjusts for double-height characters, and the printout is horizontally centered on the page. Despite the added flexibility of Screen Print II, it is also slightly faster than the original.

Screen Print II has several optimizations that improve its performance. The powers of two, used to test and set bits, are precomputed and stored in an array (make sure your graphic area is well protected from array space). Multiplication and division operations have been replaced with logical operations for added speed. Though some gains have been made in speed, don't expect miracles—printing a normal VIC screen will take about 25 minutes.

Big Screen Print

Big Screen Print is a modified version of Screen Print II that prints four dots to each screen pixel. The program has the extra complexity of mapping an even number of pixels onto odd-sized characters. This makes it twice as slow as regular Screen Print, though some may find the larger printouts worth the time. The larger format may also be more useful when printing screens with multicolor characters.

To use either program, you can include it as a subroutine in your program. Since there are no GOTOs in the code, lines can be numbered any way you wish. A better approach, for most applications, is to link programs by having your program set up the screen, do a CLR, wait for you to start

the tape recorder running, and then load the Screen Print program to start it. For example:

100 WAIT 100,1:GET A$:CLR:LOAD "SCREEN PRINT"

You should make sure the original program is larger than Screen Print to allow the program linking to work.

Both programs work no matter how the VIC's screen is shaped, no matter where it is located in memory. Before printing the screen, the programs find where the character set resides in memory, so the programs work with both ROM characters and user-definable RAM characters. They also work with mixed user and ROM characters, as well as adjust for double-height characters. Anything that can be PRINTed or POKEd onto the screen will be printable by Screen Print II and Big Screen Print.

Program 1. Screen Print II

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 V=36866:U=127:H=(PEEK(V+1)AND126)/2:W=PEEK(V)ANDU:FORI=.TO7
  :T%(7-I)=2↑I:NEXT                                    :rem 237
2 E$=CHR$(16)+MID$(STR$(40-INT(W/2)),2,2):N=PEEK(V+3)AND15:X=
  N=15                                                 :rem 162
3 T=128:S=4*(PEEK(V)ANDT)+64*(PEEK(V+3)AND112):Z=(PEEK(V+1)AN
  D1)*8+8:I=Z-1                                        :rem 247
4 J=H*Z:K=W*8-1:F=2↑15:N=(NAND7)*1024-F*((NAND8)=.):OPEN4,4:P
  RINT#4,CHR$(8)                                       :rem 73
5 FORL=.TOJ-1STEP7:PRINT#4,E$;:FORC=.TOK:B=T%(CAND7):A=T:P=S+
  C/8                                                  :rem 7
6 FORR=LTOR+6:Y%=R/Z:D=PEEK(P+Y%*W):M=N:IFXTHENIFD>UTHENM=F:D
  =D-T                                                 :rem 16
7 IFR<JTHENIFPEEK(D*Z+M+(RANDI))ANDBTHENA=A+T%(7+L-R) :rem 33
8 NEXT:PRINT#4,CHR$(A);:NEXT:PRINT#4:NEXT:CLOSE4:END :rem 148
```

Program 2. Big Screen Print

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 V=36866:U=127:H=(PEEK(V+1)AND126)/2:W=PEEK(V)ANDU:FORI=.TO7
  :T%(7-I)=2↑I:NEXT                                    :rem 237
2 E$=CHR$(16)+MID$(STR$(35-W),2,2):N=PEEK(V+3)AND15:X=N=15:F$
  =CHR$(26)+CHR$(2)                                    :rem 83
3 T=128:S=4*(PEEK(V)ANDT)+64*(PEEK(V+3)AND112):Z=(PEEK(V+1)AN
  D1)*8+8:I=Z-1                                        :rem 247
4 J=H*Z:K=W*8-1:F=2↑15:N=(NAND7)*1024-F*((NAND8)=.):OPEN4,4:P
  RINT#4,CHR$(8)                                       :rem 73
5 FORL=.TOJ-1STEP7:FORE=.TO1:PRINT#4,E$;:G=L+E*3:FORC=.TOK:B=
  T%(CAND7):A=T                                        :rem 148
6 P=S+C/8:FORR=GTOR+3:Y%=R/Z:D=PEEK(P+Y%*W):M=N:IFXTHENIFD>UT
  HENM=F:D=D-T                                         :rem 247
7 IFPEEK(D*Z+M+(RANDI))ANDBTHENY=R-G:Q=2*Y+E*(Y>.):A=A+T%(7-Q
  )-(Y<>3-3*E)*T%(7-Q-1)                               :rem 48
8 NEXT:PRINT#4,F$;CHR$(A);:NEXT:PRINT#4:NEXT:NEXT:CLOSE4:END
                                                       :rem 178
```

Pierre
Pondrom

Tape Catalog

Working with cassette tapes can be time-consuming. Not all of this time is spent waiting for the computer to load or save a desired program. Some time is spent positioning the cassette tape to the correct starting location.

Two methods have been proposed to accomplish this positioning. The first method (recommended in the user manual) is to verify the preceding program and after receiving the VERIFY ERROR, save the program right behind the last one on tape. If your cassette recorder has a counter (and you remember to reset it at the beginning of the tape), the starting position and the program name can be recorded in a notebook. One problem with this method is that a program enlarged in a revision cannot be stored at the same starting position if there is a program behind it. An enlarged program will wipe out the trailing program. What usually happens is that another entry must be made to the notebook.

Thanks to Harvey B. Herman's "Fast Find" *(COMPUTE!* magazine, July 1982) the second method allows for expansion space between programs. This utility program is stored at the beginning of each storage tape. It contains a directory of the programs stored on the tape. This program instructs the user to operate the cassette recorder in order to fast forward to the starting position where a program has been previously stored.

There are two problems with this method. Saving the program requires running the utility program to position the tape. So an updated program must be saved on a second tape temporarily in order to load the utility program. The second problem is that the space between programs becomes larger as the tape reaches the end, due to the increase in the diameter of the take-up reel and the fixed amount of time for the fast forward (F.FWD) operation. If the gap was reduced, then less time would be required to position the tape.

In order to solve both of these problems the Fast Find program was altered to specify the starting location of each program and to maintain a minimum gap between programs. To accomplish this the tape counter was calibrated with a means of relating program size to tape length.

Tape Calibration

The current version of Fast Find uses a varying time delay with the cassette in the F.FWD mode to position the tape to the beginning of one of a number of programs stored on a tape. An equation for defining the time delay for the F.FWD loop in terms of the tape counter value was developed using the Fast Find program. This timing loop was modified to include an input variable which allows the F.FWD time to vary. The table shows the F.FWD time (T)

and the associated tape counter value (C). Note at T=0, the value of C represents the length of the Fast Find program.

F.FWD Time vs. Tape Counts

T Jiffies	C Counts
0	6.0
1000	57.9
2000	122.5
3000	206.1
4000	310.1
5000	452.1

This data was then used to form the equation:

T = 104 + 16.72*C − 0.01309*C*C

In order to relate program size to tape length, a linear relationship between them was defined:

C*n* = CO + C/S * S*n*/1000

where C*n* is the count for the *n*th program;
CO is the count independent of file size;
C/S is the counter change per kilobyte of program;
S*n* is the size of the *n*th program.

Saving a short program (8 bytes) and a larger program (3574 bytes), the change in counter value was 3 and 20, respectively. Therefore, the linear constants are

CO = 3

C/S = (20−3)/(3574−8) = 4.76

Putting It All Together

Now that the tape is calibrated, a specified file size can be used to define the F.FWD time. Instead of entering the data (NAME and SIZE) as in the Fast Find program, it is more convenient to use the dynamic keyboard technique. This scheme stores data as part of the program.

The program is structured around this dynamic keyboard technique. The data is stored in four arrays (C%, A$, SN, SM). C% contains the starting tape counter value for each program stored on the tape. A$ contains the program names. SN and SM contain the program's size in kilobytes. SN is the current value while SM is the maximum value.

Each element in these arrays represents cataloged information of the programs as they are stored on the tape. However, the zero element is used for data destroyed by the dynamic keyboard. The value of C%(0) is the number of cataloged programs. SN(0) is the counter value for the next program to be stored, and SM(0) is the current size of "Tape Catalog."

Program Operation

After the DATA statements are read, a menu appears that asks if you want to add to the directory or load a program. If you select A, you are prompted for data for the next program to be added (the catalog data must be entered in order). First, you are asked the filename. It must match the one on tape. Next, you are prompted for the program size (current and maximum) in units of bytes. The kilobyte equivalent is then shown in the format used later. If you accept the data, it is added to the program by the dynamic keyboard subroutine. If not, you are returned to the beginning menu.

If you select B, the directory is displayed and you are asked to select a program number from a list assigned to each file as it was entered (same as Fast Find). An invalid number will send you back to the beginning menu. A valid number will display the time delay (in seconds) being implemented in the F.FWD mode. The remaining program operation is the same as Fast Find.

Operational Notes

The simplest method to calculate SN is to load the *n*th program and print the difference between the memory size minus FRE(0). For example, for the unexpanded VIC this would be

PRINT 3583 − FRE(0)

If the current memory size for the VIC is not known, this instruction will work:

PRINT PEEK(45) − PEEK(43) + 256*(PEEK(46) − PEEK(44))

Don't forget to save the Tape Catalog program each time after the directory has been updated. A reminder has been added. If you make a mistake in a program name or current size (SN), the offending DATA statement can be found and changed at line 10001 and up.

In order to account for inaccuracies in the tape calibration process, a program's maximum size (SM) should always be larger than the current size (SN). For the 60-minute tape used here, 210 bytes represent a tolerance of one tape count.

If you want to start a new tape, just change the value of C%(0) in line 10000 and save the program on the new tape.

Tape Catalog

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 RESTORE:REM READ # OF PROGRAMS = C%(0)          :rem 8
20 READJ:RESTORE:N=J+1                            :rem 145
25 REM TAPE CALIBRATION COEFFICIENTS               :rem 15
26 A0=104:A1=16.72:A2=-.01309:C0=6                :rem 244
```

```
30 CO=3:CS=4.76:REM SIZE COEFFICIENTS                    :rem 12
40 DIMC%(N),A$(N),SN(N),SM(N)                            :rem 76
50 NC=22:NR=23:SB=36879:REM SCREEN                       :rem 14
60 FORI=0TOJ                                            :rem 240
70 READC%(I),A$(I),SN(I),SM(I):NEXTI                     :rem 63
80 POKESB,254                                           :rem 243
90 PRINT"{CLR}{BLU}{5 SPACES}TAPE CATALOG"               :rem 17
95 PRINT"{2 DOWN}NUMBER OF PROGRAMS ON THIS TAPE = ";J
                                                        :rem 235
100 PRINT"{3 DOWN} A) ENTER PROGRAM DATA":PRINT         :rem 118
110 PRINT" B) SELECT A PROGRAM"                         :rem 231
120 PRINT"{3 DOWN}ENTER A OR B PLEASE "                 :rem 243
130 GETK$:IFK$="A"THEN500                               :rem 161
140 IFK$<>"B"THEN130                                     :rem 86
145 REM LIST DIRECTORY                                  :rem 111
150 POKESB,8                                            :rem 190
160 FORK=0TOJSTEP5                                      :rem 148
165 REM PRINT HEADER                                    :rem 182
170 PRINT"{CLR}{CYN}***PROGRAM*** SIZE-(K){YEL}NO.{WHT}***NAM
    E*** {GRN}NOW {RED}/MAX"                            :rem 129
175 REM PRINT DATA                                       :rem 40
180 GOSUB400                                            :rem 173
190 NEXTK                                                :rem 36
200 PRINT"SAVE NEXT PROGRAM{5 SPACES}AT COUNTER NUMBER "SN(0)
                                                        :rem 137
210 INPUT"{YEL}PROGRAM NO.(OR 0)";K                     :rem 144
220 IFK<1 OR K>J THEN80                                 :rem 230
230 IF(PEEK(37151)AND64)=0THENPRINT"PRESS STOP ON CASSETTE"
                                                         :rem 98
240 IF(PEEK(37151)AND64)=0THEN240                        :rem 92
250 PRINT"PRESS FAST FORWARD":PRINT"PLEASE"             :rem 253
260 IF(PEEK(37151)AND64)=64THEN260                      :rem 154
270 PRINT"THANK YOU":PRINT                              :rem 164
300 A=TI:REM SAVE TIME                                   :rem 42
301 REM D=COUNTER OFFSET                                :rem 224
302 D=SM(0)-C0:T0=A0+A1*D+A2*D↑2                        :rem 234
305 REM C=COUNTER                                        :rem 28
310 C=C%(K):T1=A0+A1*C+A2*C↑2                            :rem 42
315 REM DT=DELAY TIME                                   :rem 240
320 DT=T1-T0:S=INT(DT/60)                               :rem 211
330 PRINT"WAIT "S;"SECONDS "                            :rem 125
335 PRINT"{HOME}";:FORI=1TONR:PRINT"{DOWN}";:NEXT       :rem 165
340 IFTI-A <DTTHEN340                                   :rem 203
350 POKE37148,PEEK(37148)AND247                         :rem 231
360 PRINT"PRESS STOP PLEASE"                            :rem 247
370 IF(PEEK(37151)AND64)=0 THEN370                      :rem 100
380 PRINT"{CLR}{3 DOWN}LOAD ";CHR$(34);A$(K);CHR$(34);CHR$(19
    )                                                   :rem 109
390 POKE198,1:POKE631,13:END                            :rem 110
400 FORL=1TO5                                            :rem 13
```

```
405 REM FIVE/SREEN                                          :rem 83
410 I=L+K:IFI>JTHEN480                                      :rem 66
420 I$=MID$(STR$(I),2,2):PRINT"{YEL}"I$+" ";:IFI<10THENPRINT"
    ";                                                     :rem 172
425 REM FORMAT XX.X                                        :rem 126
430 A=SN(I):GOSUB700:SN$=K$                                 :rem 76
440 A=SM(I):GOSUB700:SM$=K$                                 :rem 75
450 PRINT"{WHT}"A$(I);TAB(13);"{GRN}"SN$;" {RED}"SM$;"{PUR}CO
    UNTER ="C%(I):PRINT                                     :rem 55
460 NEXTL                                                   :rem 37
465 PRINT"HIT SPACE BAR FOR MORE"                          :rem 176
470 GETK$:IFK$=""THEN470                                   :rem 109
480 RETURN                                                 :rem 124
490 REM INPUT DATA                                          :rem 43
500 L=N+10000:POKESB,120                                    :rem 69
510 PRINT"{CLR}{3 SPACES}PROGRAM NO.";N:PRINT               :rem 45
520 INPUT"NAME";A$(N)                                      :rem 203
530 PRINT"INPUT SIZE DATA{7 SPACES}(IN BYTES)"             :rem 189
540 INPUT"{DOWN} NOW";A:PRINT"NOW =";:GOSUB750:SN(N)=A
                                                           :rem 179
550 SN$=STR$(A)                                             :rem 75
560 INPUT"{DOWN} MAX";A:PRINT"MAX =";:GOSUB750:SM(N)=A
                                                           :rem 152
570 PRINT"IS INPUT OK(Y/N)?"                               :rem 153
580 GETK$:IFK$=""THEN580                                   :rem 113
590 IFK$="N"THEN80                                            :rem 2
600 SM$=STR$(A)                                             :rem 70
605 REM MAX SIZE OF TAPE CATALOG IS 25                      :rem 93
610 IF N=1 THEN SN(0)=25                                   :rem 215
620 C$=STR$(SN(0))                                         :rem 204
630 GOSUB650                                               :rem 180
640 GOTO670:REM DONE                                       :rem 180
645 REM SETUP DYNAMIC KEYBOARD                             :rem 106
650 O$=MID$(STR$(L),2,5)+"DATA"+C$+","+A$(N)+","+SN$+","+SM$
                                                            :rem 13
655 REM DO IT NOW                                          :rem 168
660 RL=670:GOTO900                                          :rem 32
670 PRINT"{CLR}{BLK}DON'T FORGET TO REWIND {DOWN}AND SAVE THE
     REVISED {DOWN}{4 SPACES}PROGRAM"                       :rem 79
680 IFPEEK(197)=64THEN680                                  :rem 184
690 GOTO10                                                  :rem 57
700 REM FORMAT XX.X                                        :rem 122
710 K$=STR$(A)                                             :rem 243
720 IFVAL(K$)=INT(VAL(K$))THENK$=K$+".0"                   :rem 254
730 IFLEN(K$)=3THENK$=" "+K$                               :rem 241
740 RETURN                                                 :rem 123
745 REM BYTE > KILOBYTE                                     :rem 89
750 A=A/1000                                                :rem 75
760 A%=INT((A-INT(A)+.05)*10)                                :rem 1
770 A=INT(A)+A%/10                                         :rem 186
780 PRINTA                                                 :rem 109
```

```
790 RETURN                                                    :rem 128
900 REM DYNAMIC KEYBOARD SUBROUTINE                           :rem 227
910 PRINT"{CLR}{2 DOWN}{YEL}":                                :rem 248
920 REM INVISIBLE SCREEN                                      :rem 228
930 C%(0)=N                                                    :rem 16
935 REM NEXT STORING LOCATION SN(0)                           :rem 101
940 SN(0)=SN(0)+INT(CO +CS*SM(N))                              :rem 65
945 REM CURRENT SIZE OF TAPE CATALOG                          :rem 158
950 U=(256*(PEEK(46)-PEEK(44))+PEEK(45)-PEEK(43))/1000*CS + C
    O                                                           :rem 5
960 PRINT"10000DATA"C%(0);",DIRECTORY,";SN(0)",";U:PRINTO$
                                                              :rem 135
970 PRINT"GOTO";RL:PRINT"{HOME}"                              :rem 161
980 POKE198,9                                                 :rem 215
990 FORI=0TO8:POKE631+I,13:NEXT                               :rem 154
999 END                                                       :rem 130
10000 DATA 2 ,DIRECTORY, 65 , 16.74212                        :rem 118
10001 DATA 25,COPY, 2.7, 3.6                                   :rem 96
10002 DATA 45,COPY2, 2.7, 3.6                                 :rem 149
```

Gregory
Sommerville

Microassembler

Try typing in a BASIC arcade-type game program on your VIC. How fast is it? Not very. There's only one solution for higher speed. Machine language.

Machine language is a powerful tool for computer programmers. It's faster than BASIC, and it allows you to access routines and locations you could not otherwise reach. If you've programmed only in BASIC, machine language can be hard to learn. A BASIC programmer will come upon hexadecimal numbers, offsets, and numerous addressing modes.

"Microassembler" was written to solve as many of these problems as possible. It requires a VIC with at least 8K memory expansion.

Assembler Mode

Microassembler uses a unique syntax for its assemble section. Here are the basic guidelines:

1. Single-byte instructions, like CLC or SED, can be typed as normal.
2. Zero page instructions have a Z following their root form, like ORAZ, ANDZ, DECZ, and so on.
3. Indexed instructions always have their index at the end, like ANDZX, DECX, LDXY, and so on.
4. Immediate instructions always have the symbol # at the end, for example, LDX#, CMP#, and AND#.
5. For simple absolute instructions, just use the normal form, like LDA, STX, CMP, and LSR.
6. For indirect indexing, such as ORA (addr),X use ORA(X. Always put the index at the end.
7. For an indirect jump, use JMP( ) addr.
8. For accumulator addressing, just add an *A* to the end of the code, for example, LSRA and ASLA.
9. The operation is always assumed to be the first five characters of the input string, with the operand comprising the remainder. Use enough spaces so that your operand will start at the sixth character or greater. All operation codes are from three to five characters in length.
10. Hitting RETURN while assembling simply separates the program listing on the screen. It does not affect the program.
11. To exit the Assemble mode, type DONE and press RETURN.
12. All numbers are base ten while assembling.
13. Branching instructions use the address you wish to jump to.

The program uses a decimal format simply because I feel that decimal is easier to use than hex. For example, I find it much simpler to type 175 than AF.

The rules above apply only to the Assemble mode. There are five other modes in the program.

Disassemble Mode

Typing D while in the command phase will select the Disassemble mode. An initial address will be asked for, and the program will disassemble starting there. If a location does not hold an instruction, an *X* will be printed. Branches are calculated using a routine at line 10000. It's always wise to check a program after assembling it to check for typing errors. The program will disassemble ten instructions and then wait for any key to be pressed except E. Typing E for Exit will exit to the command phase.

Hex Val/Loc.

The Hex Values of Locations mode is for those programmers who refuse to abandon hex numbers. This mode allows you to enter a series of hexadecimal digits into memory. It's also good for converting hex addresses to decimal. To enter data, simply type in your starting hex address, and then type in your data and press RETURN. The program will show you the contents of each location before you change it. Hitting RETURN alone will leave the contents as they are and go on to the next location. Type *X* followed by RETURN to exit this mode.

LOAD/SAVE RAM

Typing L for LOAD RAM will load RAM into the memory area it was saved from. Tape users should change the 8 in lines 620 and 740 to 1. If you wish a relocated LOAD, change the 1 in line 620 to 0. This will load the data into the start of memory, destroying Microassembler unless vital pointers are changed.

Typing S for SAVE RAM will save a program from any section of memory of any length. Simply type in the name, the starting memory, and the ending memory location.

Decimal Val/Loc.

The V mode, or Decimal Values of Locations mode, does exactly the same thing as the H mode, or Hex Values mode, except it works in decimal.

Learning 6502 Machine Language

Aside from this program, I would also advise you to look at some books about 6502 machine language. Get a copy of the *VIC-20 Programmer's Reference Guide*. Starting on page 107, it has a section for machine language programmers that's really helpful.

Some other books you might find helpful are *6502 Assembly Language Programming* by Lance Levanthal, *Machine Language for Beginners* by Richard

Mansfield, and *Programming the 6502* by Rodnay Zaks. Also, comprehensive memory guides are almost necessities once you get into serious machine language programming. *COMPUTE!'s Mapping the VIC* by Russ Davies is a complete memory map for the VIC, and *COMPUTE!'s VIC-20 and Commodore 64 Tool Kit*, both the BASIC and Kernal volumes, by Dan Heeb can be very useful if you want to look more closely at ROM routines.

Locating Your Programs

The Microassembler takes up memory from 4608 to 9728, leaving 6655 bytes free for your programs. The top of memory for an expanded VIC with one 8K expander is 16383.

Some Examples

Since the syntax for Microassembler is so different from that of other assemblers, I have included some short routines in standard and Microassembler format for comparison.

First, we'll write the program in standard 6502 assembly language, starting at location 16000, and then we'll see the program in the Microassembler version.

```
   3E80:  LDX  #$00
   3E82:  STX  $A2
AA 3E84:  CMP  $A2
   3E86:  BNE  AA
   3E88:  RTS
```

And now, the Microassembler version:

```
16000:LDX#  0
16002:STXZ  162
16004:CMPZ  162
16006:BNE   16004
16008:RTS
```

This program simply clears location 162 and then waits for it to equal the value of the accumulator. The machine will automatically increment location 162 by one every 1/60 second.

The Microassembler program is very close to the standard form in this particular case. Let's try another, and this one will be longer and more complex. Let's use the Kernal CHROUT subroutine. The CHROUT subroutine sends a single character to the output device, which is usually the screen. In other words, it can be used to print an ASCII character on the screen.

We'll use the subroutine to print out a string up to 255 characters long, starting at a location pointed to by the X and Y registers. The length of the string will be in the accumulator. Once again, the routine will start at location 16000 decimal. First, we'll write the standard version:

```
   3E80: STA  $3E7F
A  3E83: STX  B+01
   3E86: STY  B+02
B  3E89: LDA  $0000
   3E8C: JSR  $FFD2
   3E8F: DEC  $3E7F
   3E92: BEQ  C
   3E94: INX
   3E95: BNE  A
   3E97: INY
   3E98: JMP  A
C  3E9B: RTS
```

Now, we'll see the Microassembler version:

```
16000:STA  15999
16003:STX  16010
16006:STY  16011
16009:LDA  0
16012:JSR  65490
16015:DEC  15999
16018:BEQ  16027
16020:INX
16021:BNE  16003
16023:INY
16024:JMP  16003
16027:RTS
```

The location of the string is calculated as 256*Y + X. This is standard high byte/low byte form in 6502 machine language. The length is stored at 15999 because we need the accumulator free for the CHROUT routine at 65490, which prints the ASCII of the accumulator to the output device, which in this case is the screen.

The X and Y registers are stored into the locations used in the LDA instruction. This cuts down on the length of the routine and makes it faster and more efficient. After loading and printing, the program decrements the stored length and ends if it is equal to zero—that is, if all the characters have been printed. If the length is not zero, the program adjusts the X and Y registers, which point to the character to be printed, and jumps back to the start again, storing the address and printing as before.

Microassembler

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 POKE36879,154                                          :rem 59
6 POKE65Ø,255                                           :rem 2ØØ
1Ø PRINT"{CLR}{BLU}{RVS}{3 SPACES}MICRO ASSEMBLER{4 SPACES}"
                                                         :rem 14
```

```
20 PRINT"{3 DOWN} {RVS}COMMANDS:{DOWN}"                    :rem 21
21 PRINT" A - ASSEMBLE"                                   :rem 238
22 PRINT" D - DISASSEMBLE"                                :rem 210
23 PRINT" H - HEX VAL/LOC."                               :rem 174
24 PRINT" L - LOAD RAM"                                   :rem 176
25 PRINT" S - SAVE RAM"                                   :rem 199
26 PRINT" V - DEC VAL/LOC."                               :rem 166
27 PRINT" X - EXIT"                                       :rem 249
28 POKE52,38:POKE56,38                                      :rem 2
30 A$="0123456789ABCDEF"                                  :rem 235
40 PRINT"{DOWN}C: {LEFT}";:                               :rem 213
41 GETB$:IFB$=""THEN41                                    :rem 239
42 PRINTB$                                                 :rem 89
50 IFB$="A"THEN300                                        :rem 222
60 IFB$="D"THEN450                                        :rem 232
70 IFB$="H"THEN200                                        :rem 230
75 IFB$="L"THEN600                                        :rem 243
77 IFB$="S"THEN700                                        :rem 253
80 IFB$="V"THEN100                                        :rem 244
90 IFB$="X"THENPRINT"{CLR}":END                           :rem 219
93 IFB$=CHR$(13)ORB$=CHR$(17)THENPRINT"{UP}";              :rem 30
94 IFB$=CHR$(145)THENPRINT"{DOWN}";                       :rem 215
95 PRINT"{3 UP}":GOTO40                                   :rem 201
100 PRINT"{CLR}{RVS}{4 SPACES}DECIMAL VALUES{4 SPACES}"
                                                          :rem 198
140 INPUT"{2 DOWN} START LOC.";L:IFL<0ORL>65535THEN140:rem 36
150 PRINTL;"--";                                          :rem 131
160 PRINTPEEK(L);"--";                                    :rem 250
165 HX$=""                                                :rem 225
170 INPUTHX$:IFHX$="X"THENCLR:GOTO10                       :rem 54
175 IFHX$<>""THENPOKEL,VAL(HX$)                           :rem 124
180 L=L+1:IFL>65535THEN140                                :rem 233
190 GOTO150                                               :rem 105
200 PRINT"{CLR}{RVS}{6 SPACES}HEX{2 SPACES}VALUES{5 SPACES}"
                                                          :rem 189
205 T=0                                                    :rem 88
210 INPUT"{3 DOWN} START LOC.";L$                         :rem 223
220 HX$=LEFT$(L$,2):GOSUB1000                             :rem 126
230 T=VAL(HX$)*256                                        :rem 229
240 HX$=RIGHT$(L$,2):GOSUB1000                            :rem 211
250 T=T+VAL(HX$)                                          :rem 159
260 IFT<=65535THENPRINTT;"--";:GOTO270                     :rem 44
265 GOTO205                                               :rem 109
270 G=PEEK(T):HX$=STR$(G):GOSUB2000                        :rem 83
280 PRINTHX$;"--";                                        :rem 255
285 HX$=""                                                :rem 228
287 INPUTHX$:IFHX$=""THENT=T+1:GOTO260                    :rem 126
288 IFHX$="X"THENCLR:GOTO10                               :rem 178
290 GOSUB1000:POKET,VAL(HX$):T=T+1                         :rem 56
295 GOTO260                                               :rem 113
300 PRINT"{CLR}{RVS}{7 SPACES}ASSEMBLE{7 SPACES}"          :rem 85
```

```
310 INPUT"{3 DOWN}START LOC.";L                          :rem 188
320 PRINT"{3 DOWN}"                                      :rem 153
330 PRINT"{LEFT}"L":";:GOSUB2100:IFU$="DONE"THEN10       :rem 255
333 IFU$=""THEN330                                       :rem 231
335 IFLEN(U$)<6THENU$=U$+" ":GOTO335                      :rem 32
340 CD$=LEFT$(U$,5):N$=RIGHT$(U$,LEN(U$)-5)              :rem 133
347 RESTORE                                              :rem 194
350 FORG=0TO255                                          :rem 114
360 READNM$                                              :rem 116
361 IFNM$="X"THEN370                                     :rem 138
362 IFLEN(NM$)<6THENNM$=NM$+" ":GOTO362                  :rem 242
365 IFCD$=RIGHT$(NM$,5)THEN380                           :rem 242
370 NEXT:FORTH=1TO500:POKE36878,15:POKE36876,248:NEXT:POKE368
    78,0:POKE36876,0:GOTO330                             :rem 178
380 POKEL,G:IFLEFT$(NM$,1)="1"THENL=L+1:GOTO330           :rem 41
385 IFLEFT$(CD$,1)="B"THEN11000                          :rem 187
390 IFLEFT$(NM$,1)="3"THEN410                             :rem 95
400 POKEL+1,VAL(N$):L=L+2:GOTO330                        :rem 178
410 F1=VAL(N$):F2=INT(F1/256):F3=F1-F2*256                :rem 48
420 POKEL+1,F3:POKEL+2,F2                                :rem 200
430 L=L+3:GOTO330                                        :rem 211
450 PRINT"{CLR}{RVS}{5 SPACES}DISASSEMBLE{6 SPACES}"      :rem 59
460 INPUT"{3 DOWN}START LOC.";L                          :rem 194
470 PRINTL;"-";                                           :rem 91
480 RESTORE:FORG=0TOPEEK(L)                              :rem 250
490 READNM$:NEXT                                         :rem 241
500 PRINT"{RVS}";MID$(NM$,2);:E=VAL(LEFT$(NM$,1))          :rem 6
503 IF MID$(NM$,2,1)="B"THEN9000                         :rem 171
505 IFNM$="X"THENPRINT"X":L=L+1:E=0:GOTO540              :rem 124
510 IFE=1THENPRINT:GOTO540                               :rem 160
520 IFE=2THENPRINTPEEK(L+1):GOTO540                      :rem 192
530 T1=PEEK(L+1):T2=PEEK(L+2):PRINT256*T2+T1              :rem 88
540 GV=GV+1:IFGV/10=INT(GV/10)THEN560                    :rem 210
550 L=L+E:GOTO470                                        :rem 237
560 GETZ$:IFZ$=""THEN560                                 :rem 139
565 IFZ$="E"THEN10                                         :rem 3
567 PRINT:L=L+E                                          :rem 174
570 GOTO470                                              :rem 112
600 PRINT"{CLR}{RVS}{7 SPACES}LOAD RAM{7 SPACES}"         :rem 12
610 INPUT"{3 DOWN}PROGRAM NAME";PN$                      :rem 212
620 LOADPN$,8,1                                           :rem 59
700 PRINT"{CLR}{RVS}{7 SPACES}SAVE RAM{7 SPACES}"         :rem 28
705 INPUT"{3 DOWN}PROGRAM NAME";PN$                      :rem 217
710 NL=LEN(PN$)                                           :rem 97
715 INPUT"{2 DOWN}START MEM";SM                          :rem 219
720 INPUT"{2 DOWN}END MEM";EM:EM=EM+1                      :rem 9
725 IFSM>EMOREM<0OREM>65535ORSM<0ORSM>65534THEN705       :rem 179
730 S1=INT(SM/256):S2=SM-S1*256:POKE172,S2:POKE173,S1:rem 127
735 POKE193,S2:POKE194,S1:E1=INT(EM/256):E2=EM-E1*256 :rem 68
740 POKE174,E2:POKE175,E1:POKE186,8:POKE183,LEN(PN$)  :rem 64
745 FORJ=1TOLEN(PN$):POKE819+J,ASC(MID$(PN$,J,1)):NEXT :rem 5
```

```
750 POKE187,52:POKE188,3:POKE185,1                    :rem 203
760 INPUT"{2 DOWN}{RVS}HIT RETURN TO SAVE";P$         :rem 235
775 SYS63109:CLR:RUN                                  :rem 239
999 END                                               :rem 130
1000 TM=0:FORJ=1TO2                                   :rem 125
1010 FORI=0TO15                                       :rem 104
1020 IFMID$(A$,I+1,1)=MID$(HX$,J,1)THENTM=TM+I*16↑(2-J)
                                                       :rem 98
1030 NEXTI,J:HX$=STR$(TM):RETURN                       :rem 38
2000 HX=VAL(HX$):D1=HXAND240                          :rem 140
2010 D2=HXAND15:D1=D1/16                               :rem 70
2020 HX$=MID$(A$,D1+1,1)+MID$(A$,D2+1,1)               :rem 13
2030 RETURN                                           :rem 165
2100 U$="":US$=""                                      :rem 68
2200 U$="":GETU$:IFU$=""THEN2200                        :rem 7
2212 POKE36878,15:POKE36876,242:FORG1=1TO35:NEXT:POKE36878,0
                                                       :rem 51
2214 IFU$=CHR$(13)THEN2230                            :rem 186
2215 PRINT"{RVS}";U$;                                 :rem 156
2217 IFU$=CHR$(20)THENUS$=LEFT$(US$,LEN(US$)-1):U$="" :rem 35
2220 US$=US$+U$                                        :rem 63
2225 GOTO2200                                         :rem 200
2230 U$=US$:PRINT:RETURN                               :rem 42
3000 REM*** DATA FOR{3 SPACES}ASSEMBLING/DISASSEM-{2 SPACES}B
     LING                                              :rem 44
3010 DATA1BRK,2ORA(X,X,X,X,2ORAZ,2ASLZ,X,1PHP,2ORA#,1ASLA
                                                       :rem 22
3020 DATAX,X,3ORA,3ASL,X,2BPL,2ORA(Y,X,X,X,2ORAZX      :rem 58
3030 DATA2ASLZX,X,1CLC,3ORAY,X,X,X,3ORAX,3ASLX,X       :rem 49
3040 DATA3JSR,2AND(X,X,X,2BITZ,2ANDZ,2ROLZ,X,1PLP,2AND#,1ROLA
     ,X                                               :rem 179
3050 DATA3BIT,3AND,3ROL,X,2BMI,2AND(Y,X,X,X           :rem 106
3060 DATA2ANDZX,2ROLZX,X,1SEC,3ANDY,X,X,X,3ANDX,3ROLX,X
                                                       :rem 28
3070 DATA1RTI,2EOR(X,X,X,X,2EORZ,2LSRZ,X,1PHA,2EOR#,1LSRA,X
                                                      :rem 207
3080 DATA3JMP,3EOR,3LSR,X,2BVC,2EOR(Y,X,X,X,2EORZX,2LSRZX,X
                                                       :rem 29
3090 DATA1CLI,3EORY,X,X,X,3EORX,3LSRX,X,1RTS,2ADC(X,X,X,X
                                                      :rem 106
3100 DATA2ADCZ,2RORZ,X,1PLA,2ADC#,1RORA,X,3JMP(),3ADC,3ROR,X
                                                      :rem 141
3110 DATA2BVS,2ADC(Y,X,X,X,2ADCZX,2RORZX,X,1SEI,3ADCY,X,X,X
                                                      :rem 216
3120 DATA3ADCX,3RORX,X,X                              :rem 229
3130 DATA2STA(X,X,X,2STYZ,2STAZ,2STXZ,X,1DEY,X,1TXA,X:rem 167
3140 DATA3STY,3STA,3STX,X,2BCC,2STA(Y,X,X,2STYZX,2STAZX,2STXZ
     Y,X                                              :rem 207
3150 DATA1TYA,3STAY,1TXS,X,X,3STAX,X,X,2LDY#,2LDA(X,2LDX#,X
                                                      :rem 179
3160 DATA2LDYZ,2LDAZ,2LDXZ,X,1TAY,2LDA#,1TAX,X        :rem 113
```

```
3170 DATA3LDY,3LDA,3LDX,X,2BCS,2LDA(Y,X,X,2LDYZX,2LDAZX,2LDXZ
     Y,X                                                  :rem 65
3180 DATA1CLV,3LDAY,1TSX,X,3LDYX,3LDAX,3LDXY,X           :rem 177
3190 DATA2CPY#,2CMP(X,X,X,2CPYZ,2CMPZ,2DECZ,X,1INY,2CMP#,1DEX
     ,X                                                   :rem 162
3200 DATA3CPY,3CMP,3DEC,X,2BNE,2CMP(Y,X,X,X,2CMPZX,2DECZX,X
                                                          :rem 186
3205 DATA1CLD,3CMPY,X,X,X                                  :rem 12
3210 DATA3CMPX,3DECX,X,2CPX#,2SBC(X,X,X,2CPXZ,2SBCZ,2INCZ,X,1
     INX,2SBC#,1NOP,X                                      :rem 56
3220 DATA3CPX,3SBC,3INC,X,2BEQ,2SBC(Y,X,X,X,2SBCZX,2INCZX,X,1
     SED,3SBCY,X,X,X                                       :rem 23
3230 DATA3SBCX,3INCX,X                                     :rem 90
9000 IFMID$(NM$,2,3)="BRK"ORMID$(NM$,2,3)="BIT"THEN505:rem 24
10000 E=PEEK(L+1):IFE=<127THENGG=L+E+2:GOTO10010           :rem 42
10005 GG=L-254+E                                           :rem 69
10010 PRINTGG:E=2:GOTO540                                   :rem 7
11000 IFLEFT$(CD$,3)="BIT"ORLEFT$(CD$,3)="BRK"THEN390       :rem 1
11010 GG=VAL(N$):IFGG>LTHENGA=GG-L-2:GOTO12000              :rem 5
11020 GA=254-L+GG                                          :rem 134
12000 POKEL+1,GA                                           :rem 126
12010 L=L+2:GOTO330                                         :rem 47
```

Royan
Webb

Special Characters in the Expanded VIC

Generating and using custom character sets on the VIC is fun. Unfortunately, their use seems to be restricted to the unexpanded machine. This is because the VIC chip cannot "see" beyond memory location 7680, the top of unexpanded memory. This greatly limits the length of BASIC programs.

If you have added 8K or more of memory, you can still POKE the character set by using POKE 36869,207 instead of 36869,255. To return to the original set, use 192 instead of 240. Unfortunately, this results in less memory for BASIC than not using the expansion memory.

I have written a short program that allows character sets (yes, you can program more than one) and still uses all memory after screen memory. The trick is to move the screen memory to a location above the character set. Then, the start of BASIC has to be set to the end of screen memory. Using Jim Butterfield's alternate screen program, I set the screen to memory location 4608. I then changed the start-of-BASIC pointers to memory location 5120 and POKEd 5120 with 0. Once these moves were made, I could use the memory from 4096 through 4607 for the character set, and then POKE 36869,204 to use the characters I had made.

After running this program to initialize the VIC, you can load any program, but remember that the screen now starts at 4608. The color memory stays at 38400, and your character set starts at 4096.

Using the Program

First, type in and save Program 1 before running, or it will self-destruct even if typed in correctly! Now run it. The program will tell you if your DATA statements are correct. When it runs correctly, you will have a screenful of garbage. Simply hit RUN/STOP–RESTORE. The VIC is now initialized. Use the program as the first portion of a double load to allow the use of special characters with any length BASIC program.

As for using multiple character sets, the screen and BASIC may be moved up again to allow as many as four character sets. To change the BASIC loader to allow this, you must change two of the numbers in line 60, three in line 65, and the checksum in line 20. The numbers to be changed in line 60 are the second and the last. The ones in line 65 are the third, sixth, and last (the number 20 in each case.) Use the following table to decide what number you wish.

For	Change to	Checksum	POKE 36869 to
2 sets	22	2465	205
3 sets	26	2485	206
4 sets	30	2505	207

The second set will start at 5120, the third at 6144, and the fourth at 7168. BASIC will start 512 bytes above the highest character set.

There is a small penalty if you use more than two character sets. Sets 3 and 4 each have a block of 512 bytes below them that can't be used by BASIC or the character sets. However, you can use these areas for two more screens or for machine language subroutines (perhaps a joystick routine). There is also a bonus: Using reverse video to print a character will cause the equivalent character from the next higher set to appear. For example, if the *A* in set 1 is changed to a smiling face, and the *A* in set 2 to a frowning face, when set 1 is in use, typing A will print a smiling face, and a reverse *A* will print a frowning face.

Program 2 will show you a familiar face. After running it, call up your free bytes and gloat over the amount of memory you have left.

Program 1. Characters

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 REM BASIC LOADER FOR 8K GRAPHICS                          :rem 233
5 PRINT"{CLR}"                                              :rem 153
10 D=0:FORX=5500TO5531:READA:POKEX,A:D=D+A:NEXT             :rem 172
20 IFD<>2455THENPRINT"ERROR IN DATA":END                    :rem 203
30 C=0:FORX=5200TO5235:READA:POKEX,A:C=C+A:NEXT             :rem 169
40 IFC<>5288THENPRINT"ERROR IN DATA":END                    :rem 211
50 SYS5500                                                   :rem 46
60 DATA169,20,133,44,133,46,133,48,133,50,141,130,2,169,1,141
   ,129,2,169,0,141,0,20                                    :rem 210
65 DATA141,1,20,141,2,20,76,80,20                           :rem 111
70 DATA169,150,141,2,144,169,18,141,136,2,162,0,169,146,232,1
   57,216,224,12,208                                         :rem 45
80 DATA250,169,147,232,157,216,224,24,208,250,169,147,32,210,
   255,0                                                    :rem 229
```

Program 2. Special Effects

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 GOTO 10                                                   :rem 203
4 PRINT"{CLR}";:POKE36869,204:FORY=1TO7:FORX=1TO7:PRINTA$;"
  {2 UP}";:NEXTX:PRINT"{RIGHT}{2 DOWN}";:NEXT               :rem 104
5 GETB$:IFB$=""THEN5                                        :rem 143
6 POKE36869,192:PRINT"{CLR}BYE, BYE NOW.":END               :rem 250
10 FORX=4096TO4607:POKEX,PEEK(X+28672):NEXT                 :rem 140
```

```
20 FORX=4136TO4143:READA:POKEX,A:NEXT:FORX=4200TO4223:READA:P
   OKEX,A:NEXT                                          :rem 129
25 FORX=4240TO4247:READA:POKEX,A:NEXT                    :rem 95
30 FORX=4256TO4263:READA:POKEX,A:NEXT:FORX=4280TO4287:READA:P
   OKEX,A:NEXT                                          :rem 154
40 FORX=4296TO4303:READA:POKEX,A:NEXT:FORX=4352TO4359:READA:P
   OKEX,A:NEXT                                          :rem 154
50 DATA193,0,0,0,0,36,36,0,36,16,8,4,4,4,4,2,2,1,1,0,0,0,0,0,
   66,60,0,129,126,0,0,0                                :rem 135
60 DATA0,129,66,60,0,0,195,102,0,7,8,16,32,32,33,35,64,128,12
   8,0,0,0,0,0,0,224,16,8                               :rem 233
70 DATA4,4,132,196,36,8,16,32,32,32,32,64                :rem 9
80 PRINT"{CLR}WHEN YOU WISH TO SEE{2 SPACES}THE SPECIAL EFFEC
   TS{3 SPACES}JUST PRESS F1"                           :rem 226
85 PRINT"{DOWN}WHEN YOU WISH TO STOP THE PROGRAM PRESS F7"
                                                        :rem 165
90 GETB$:IFB$=""THEN90                                  :rem 247
100 A$="TRY{DOWN}{3 LEFT}ME {DOWN}{3 LEFT}NOW"          :rem 204
110 PRINT"{CLR}{8 DOWN}{9 RIGHT}"A$                     :rem 232
120 GETB$:IFB$="{F1}"THENGOSUB1000                      :rem 126
130 IFB$="{F7}"THENF=1:GOTO4                            :rem 28
140 GOTO120                                             :rem 97
1000 IFPEEK(36869)=192THENPOKE36869,204:RETURN          :rem 249
1010 IFPEEK(36869)=204THENPOKE36869,192:RETURN          :rem 250
```

Todd
Wilson

Faster BASIC

One minor annoyance programmers face when working with the VIC is reading the joystick. The designers could at least have put all the inputs in a single memory location instead of scattering them all over the place. A BASIC command would have been nice, too.

Another problem is the screen manipulation. PEEKs and POKEs are too slow and cumbersome for really fast and efficient games. Again, another BASIC command or two would have been useful.

"Joyfast" and "Fasplot" make use of the VIC USR function. Fasplot also uses the USR function for looking at the screen, but it uses the SYS command in place of POKE.

In the BASIC ROM, there are subroutines to get and convert numbers from BASIC lines to machine language. Location 56475 ($DC9B) takes the value in the floating-point accumulator and converts it to an integer number. Its complement location, 54161 ($D391), converts the A and Y registers to floating-point. Location 52989 ($CEFD) checks for a comma, location 52638 ($CD9E) gets the number after the comma, and location 55287 ($D7F7) converts the number to a positive whole number from 0 to 65535.

Program 1 is Joyfast. USR(0) returns the fire button status. A one means the fire button is depressed, and a zero means that it is up. USR(1) returns the joystick direction from 0 to 8:

USR(1)	Direction
0	Center
1	N
2	NE
3	E
4	SE
5	S
6	SW
7	W
8	NW

Program 2 is Fasplot. USR(0),x,y returns the character number at column x, row y. USR(1),x,y returns the color of the location at column x, row y. And SYS address,x,y,i,j puts the character number in variable *i* at column x, row y, and colors it with the color number in variable *j*. The program displays the address to use based on your particular memory configuration.

A Bug

Because I wanted to conserve as much memory as possible, one bug remains. Whenever you use the SYS, the USR(0), or the USR(1) function in Fasplot, the cursor location is altered to the location looked at or plotted to. This may

not be much of a problem, but it can upset programmers when they are trying to write a game and the score keeps flying all over the screen.

Program 1. Joyfast

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 A=PEEK(55)+PEEK(56)*256-100:A%=A/256:POKE56,A%:POKE55,A-A%*
  256:CLR                                                  :rem 37
2 POKE0,76:POKE1,PEEK(55):POKE2,PEEK(56):Q=PEEK(1)+PEEK(2)*25
  6:H=Q+68:L=H+16                                         :rem 199
3 HH=INT(H/256):HL=H-HH*256:LH=INT(L/256):LL=L-LH*256:rem 105
10 DATA120,32,155,220,165,101,208,15                       :rem 4
20 DATA173,17,145,162,0,160,0,41                          :rem 66
30 DATA32,208,43,160,1,208,39,201                        :rem 120
40 DATA1,208,39,169,127,141,34,145                       :rem 186
50 DATA173,32,145,42,42,41,1,133                          :rem 70
60 DATA255,169,255,141,34,145,173,17                      :rem 38
70 DATA145,74,41,14,5,255,170,188                        :rem 142
80 DATAHL,HH,189,LL,LH,170,138,32                         :rem 65
90 DATA145,211,88,96                                      :rem 20
100 FORJ=QTOQ+67:READA$:A=VAL(A$):IFA$="HL"THENA=HL      :rem 108
110 IFA$="HH"THENA=HH                                    :rem 212
120 IFA$="LL"THENA=LL                                    :rem 229
130 IFA$="LH"THENA=LH                                    :rem 222
140 POKEJ,A:NEXT                                         :rem 244
150 FORJ=0TO15:READA                                     :rem 212
160 POKEJ+H,A:POKEJ+L,0:NEXT                             :rem 239
170 DATA .,.,.,6,.,8,.,7,.,.,4,5,2,1,3,0                  :rem 92
```

Program 2. Fasplot

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
0 A=PEEK(55)+PEEK(56)*256-130:POKE56,A/256:POKE55,A-PEEK(56)*
  256:CLR                                                  :rem 95
1 S=PEEK(55)+PEEK(56)*256:XX%=PEEK(56):XX=PEEK(55):Y=S+12:YY%
  =Y/256:YY=Y-YY%*256                                      :rem 25
2 X=S+45:POKE0,76:POKE2,X/256:POKE1,X-PEEK(2)*256       :rem 133
10 DATA32,253,206,32,158,205,32,247                      :rem 226
20 DATA215,165,101,96,32,126,29,165                      :rem 234
30 DATA101,201,22,16,21,133,254,32                       :rem 155
40 DATA126,29,165,101,201,23,16,10                       :rem 164
50 DATA133,255,164,254,170,24,32,240                      :rem 21
60 DATA255,96,76,72,210,120,32,155                       :rem 187
70 DATA220,162,0,165,101,240,2,162                       :rem 161
80 DATA120,134,253,32,138,29,165,209                      :rem 30
90 DATA133,247,165,210,101,253,133,248                   :rem 123
100 DATA164,211,177,247,166,253,240,2                     :rem 73
110 DATA41,15,168,169,0,32,145,211                       :rem 173
120 DATA88,96,120,32,138,29,32,126                       :rem 189
```

```
130 DATA29,165,101,133,252,32,126,29                    :rem 20
140 DATA165,101,133,253,165,209,133,243                 :rem 169
150 DATA165,210,105,120,133,244,164,211                 :rem 156
160 DATA165,252,145,209,165,253,145,243                 :rem 184
170 DATA88,96                                           :rem 189
180 FORJ=STOS+129:READA:POKEJ,A:NEXT                    :rem 71
190 POKES+14,XX%:POKES+13,XX:POKES+25,XX%:POKES+24,XX:POKES+9
    6,XX%:POKES+95,XX                                   :rem 212
200 POKES+103,XX%:POKES+102,XX:POKES+61,YY%:POKES+60,YY
                                                        :rem 166
210 PRINT"SYS"S+90",X,Y,CHR,COL"                        :rem 170
```

Chapter Three

Sound and Graphics

James Calloway

Paddle Graphics

When I was a child, one of my favorite toys was a plastic drawing board. It had two knobs, one for horizontal and one for vertical, and they controlled the movement of a stylus beneath the board's clear plastic "screen." Now the VIC-20 can be programmed to do much the same thing. Game paddles serve the function of horizontal and vertical controls, and the VIC has the added advantage of being able to draw (on a television screen) in color.

I first tried writing such a program in BASIC, and it worked beautifully except for one major flaw: The human hand is faster than BASIC. Unless the paddles were turned very slowly, the program would skip across the screen, leaving a trail of widely spaced dots rather than a solid line. Machine language was the logical solution.

Another problem cropped up once I began to get the program working in machine language. The two analog-to-digital converters in the VIC are not extraordinarily stable. Since those A/D converters are what makes the paddles work, that means the paddles are pretty jumpy.

Reading the Paddle

The flicker problem can be reduced through programming. The "Paddle Graphics" program does this by reading each paddle 128 times, adding the values together, and then dividing the sum by 128 to get an average value.

The fact that 256 additions are performed in a fraction of a second should give you an idea of how fast machine language operates. The more readings that are averaged, the less flicker there will be. However, there is a greater chance of the program skipping across the screen, because the averaging slows the program slightly. Taking 128 readings on each paddle is essentially a compromise between the two concerns. Also, it is no accident that the number of readings is a power of two. That greatly simplifies the division. To divide a binary number by 128, its bits simply are shifted seven places to the right.

Using Paddle Graphics

When the program is run, it first displays its name, "Paddle Graphics." Then the screen shrinks to about half its usual area and is filled with multicolor garbage. You can clear the screen by pressing the RETURN key. Now you are ready to draw. You have four colors to chose from: background white, which is activated by the f1 key; cyan, f3; purple, f5; and green, f7. Cyan and purple may seem like strange colors, but there are reasons for them. If nothing appears when you move the paddles, try changing the colors by

pressing the f1 key. Cyan is used because it is the normal border color. Purple serves as a brighter alternative to the VIC's somewhat dull red. You can give yourself a different selection of colors by modifying the machine language program. We'll get into that later.

Entering the Program

Paddle Graphics is in the form of a BASIC loader program. First, unplug any memory expanders you might have. Next, enter the following line before entering the program:

POKE 44,18:POKE 4608,0:NEW

If you forget to enter these POKEs, don't worry. Just finish typing and save the program. Then enter the POKEs and reload.

Now enter the program. *Save it before you run it!* When you run the BASIC loader, it will check its DATA statements for errors before POKEing the numbers into memory. If there are any errors, their line numbers will be listed on the screen, and the loader will stop so that you can correct those lines. (Lines 80 and 90 are treated as one line. An error in either will produce "80–90".) If there are no mistakes in the DATA statements, the loader will announce DATA OK—STORING and give you a READY when it finishes. If, instead, you get SYNTAX ERROR IN 70, you have forgotten to enter the line above.

Once you get it right, the loader will transform itself into a new program. LIST will reveal three lines of BASIC you've never seen before. Save the new program. Now, Paddle Graphics is ready to run. As a side effect of the BASIC loader, it may draw in two colors at once, but as soon as you hit a function key, that will be corrected.

Changing Colors

If you want to fiddle with the color choices, turn to the screen and border color chart on page 134 in the computer manual (or page 265 in the *Programmer's Reference Guide*). The screen and border colors (f1 and f3, respectively) are independent of the Paddle Graphics program. You set them by POKEing a number from the chart into location 36879. You can pick from 16 colors for the screen and 8 for the border. For example, an orange screen with a black border is 136. Stopping the Paddle Graphics program with the RUN/STOP key will reset the screen to its normal white with cyan border and will allow you to POKE 36879 with your color choice.

The other two colors, the character color (f5) and the auxiliary color (f7), are controlled by the program. To change them you will have to modify the program, but fortunately that is only a matter of making a few POKEs once you have loaded it.

There are eight character colors, the same eight as for the border, and they are indicated by the numbers 0–7. If you look at the color chart again, the border colors across the top begin with black (0) and end with yellow (7). Add 8 to that number and POKE the sum into 4284. Add 512 to that address if you have 8K expansion or greater. Subtract 3072 for 3K expansion. For example, to get yellow with 8K, POKE 4796,15.

The auxiliary color can be any of the 16 colors also available to the screen. Black, at the top of the list, is again 0. Light yellow, at the bottom, is 15. Multiply that number by 16 and POKE the product into 4305 or the equivalent location. Pink with 8K would require POKE 4817,160.

Paddle Graphics

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}CHECKING DATA":CLR:V=0                          :rem 51
20 S=0:FORQ=0TO24:READR:S=S+R:NEXTQ:READR:IFS<>RTHENPRINT"80
   {SPACE}- 90":V=1                                          :rem 162
30 FORZ=0TO31:W=0:FORY=0TO15:READX:W=W+X:NEXT:READX:IFX<>WTHE
   NPRINT5000+Z*10,:V=1                                      :rem 48
40 NEXT:IFV=1THENSTOP                                        :rem 165
50 CLR:PRINT"DATA OK - STORING"                              :rem 88
60 FORS=828TO852:READR:POKES,R:NEXTS:READX                   :rem 29
70 FORZ=0TO31:FORY=0TO15:READX:POKE4096+Z*16+Y,X:NEXTY:READX:
   NEXTZ                                                     :rem 103
80 DATA162,1,160,16,134,43,132,44,162,246,160,17,134,45,132
                                                             :rem 122
90 DATA46,134,47,132,48,134,49,132,50,0,2360                 :rem 165
100 SYS828                                                   :rem 50
5000 DATA0,62,16,10,0,153,34,147,8,17,17,17,17,17,17,17,549
                                                             :rem 125
5010 DATA29,29,29,29,80,65,68,68,76,69,32,71,82,65,80,72,944
                                                             :rem 235
5020 DATA73,67,83,34,58,153,34,17,29,29,29,66,89,32,74,65,932
                                                             :rem 22
5030 DATA77,69,83,32,67,65,76,76,79,87,65,89,34,0,77,16,992
                                                             :rem 204
5040 DATA20,0,129,83,178,48,164,55,48,48,58,130,0,104,16,30,1
     111                                                     :rem 121
5050 DATA0,158,40,194,40,52,51,41,170,194,40,52,52,41,172,50,
     1347                                                    :rem 165
5060 DATA53,54,170,49,48,57,41,0,0,0,0,0,0,0,169,11,652
                                                             :rem 167
5070 DATA141,0,144,169,37,141,1,144,173,2,144,41,128,9,16,141
     ,1431                                                   :rem 212
5080 DATA2,144,169,32,141,3,144,173,5,144,41,240,9,13,141,5,1
     406                                                     :rem 112
5090 DATA144,169,0,160,0,174,136,2,132,251,134,252,145,251,20
     0,24,2174                                               :rem 151
```

```
5100 DATA105,1,24,192,0,208,245,24,160,0,173,2,144,41,128,162
     ,1609                                                  :rem 202
5110 DATA150,201,128,240,2,162,148,134,252,162,0,169,12,145,2
     51,200,2356                                            :rem 244
5120 DATA24,208,250,232,230,252,224,2,208,243,24,173,14,144,4
     1,15,2284                                              :rem 149
5130 DATA9,80,141,14,144,32,225,255,208,3,32,210,254,32,228,2
     55,2122                                                 :rem 54
5140 DATA201,13,208,27,160,0,162,20,132,251,134,252,169,0,145
     ,251,2125                                              :rem 139
5150 DATA24,200,208,250,232,230,252,24,224,28,208,242,24,144,
     214,56,2560                                            :rem 251
5160 DATA233,133,144,8,24,201,4,176,3,141,60,3,24,162,0,134,1
     450                                                    :rem 100
5170 DATA251,134,252,134,253,134,254,24,173,8,144,101,251,133
     ,251,144,2641                                          :rem 102
5180 DATA2,230,252,24,173,9,144,101,253,133,253,144,2,230,254
     ,24,2228                                               :rem 103
5190 DATA232,224,128,208,226,102,252,102,251,24,102,254,102,2
     53,24,232,2716                                         :rem 140
5200 DATA224,135,208,241,24,144,2,144,140,165,251,133,251,24,
     106,106,2298                                            :rem 46
5210 DATA106,106,41,15,133,252,165,251,24,106,41,6,133,251,16
     5,253,2048                                             :rem 202
5220 DATA73,255,133,253,24,106,106,106,106,41,15,133,254,165,
     253,24,2047                                              :rem 2
5230 DATA106,41,6,133,253,24,169,0,162,0,6,252,232,42,24,224,
     1674                                                   :rem 162
5240 DATA7,208,247,24,105,20,141,62,3,169,0,162,0,24,6,254,14
     32                                                      :rem 60
5250 DATA232,144,1,42,24,224,3,208,245,24,109,62,3,141,62,3,1
     527                                                    :rem 110
5260 DATA24,144,2,144,162,165,251,101,254,144,3,238,62,3,24,1
     01,1822                                                 :rem 52
5270 DATA252,144,3,238,62,3,133,251,173,62,3,133,252,234,162,
     0,2105                                                   :rem 4
5280 DATA169,192,228,253,240,5,232,106,24,144,247,73,255,160,
     0,49,2377                                              :rem 189
5290 DATA251,145,251,200,145,251,173,60,3,24,106,106,106,24,1
     62,0,2007                                              :rem 143
5300 DATA228,253,240,5,232,106,24,144,247,160,0,17,251,145,25
     1,200,2503                                             :rem 194
5310 DATA145,251,24,144,174,0,0,0,0,0,0,0,0,0,0,0,738 :rem 34
```

Ritchie
Falk

VIC Graph

Many VIC-20 users want to use custom characters in their programs, but there seems to be a lot of confusion over how to handle them. My interest in custom characters is in their use for graphic displays of numeric data. This involves a process known as bitmapping, where each pixel (dot) on the screen (or portion of the screen) may be turned on or off individually.

In the VIC-20, the video chip can get its character data from several areas of memory. The address of this data is controlled by the register at 36869 (lower four bits). To use custom characters, it must be set for an area of RAM. The starting address for the data may be 4096, 5120, 6144, or 7168 as well as several ROM addresses.

The position of the screen in memory is controlled by two registers—36869 (upper four bits) and 36866 (bit 7). The screen may be placed anywhere between 4096 and 8191 in 512-byte increments. This means that there are about 64 possible combinations of character/screen placement, although some of these will result in strange screen displays. The corresponding POKEs are detailed in the *VIC Programmer's Reference Guide.* For everything to work properly, the screen, characters, and program must be in separate areas of memory.

To be sure that BASIC doesn't change the screen or character data when using custom characters, it is necessary to change the start-of-BASIC pointer (43–44), or the top-of-strings pointer (51–52) and top-of-memory pointer (55–56). With the 8K (or 16K) expander, the screen is placed at 4096 on power-up with the start of BASIC at 4609. For custom characters, I place the character data at 4096 and move the screen to 5120 (for 64 characters) or 5632 (for 128) or 6144 (for 256). The start of BASIC must then be set above the end of the screen. Note that the start-of-BASIC pointer must be changed *before* the program is loaded.

My solution to this is to record a short program with the required POKEs immediately before my graphics program. Program 1, which requires 8K or more memory expansion, will move the screen to 6144, and the start of BASIC to 6657, leaving space to program 256 characters (2K). Line 10 moves BASIC and line 20 moves the screen.

Program 1 erases itself when run so be sure to save a copy before running. After generating your custom characters, POKE 36869,236 to use them on the screen, or POKE 36869,224 to return to the regular characters. I usually copy some of the ROM characters into RAM—the first 64 include the letters, numbers, and punctuation—so that I can display graphics and regular text at the same time.

The Start of BASIC

A word about that POKE in line 10: In order for a program to run, the byte immediately preceding the start of BASIC must contain zero. On power-up with, say, the 8K expander, the start of BASIC is at 4609. There will always be a zero at 4608. If the start of BASIC is changed, that zero must be POKEd into the new location or a program will not run, although it will list. To demonstrate this, load a program and POKE 4608,1 (or any other number except zero). The program will still list, but a RUN will result in a SYNTAX ERROR. Now, POKE 4608,0 and the program will again run normally. (For the unexpanded VIC, it's 4096.)

With the unexpanded VIC (or 3K expander), it is not necessary to change the start of BASIC, since the screen and character data may be placed at the top of memory. The POKEs to change the top of memory and top of strings may now be placed in the main program.

Not All or Nothing

One misconception is that creating custom characters is an all-or-nothing situation. By setting the character data pointer to 7168 and the screen at 6656, it is possible to have 128 custom characters and still use the normal ROM characters without copying them into RAM. Printing reversed characters will now print the normal (unreversed) character set. This technique allows graphics and regular text in the same display without using up too much of the VIC's limited memory.

Using the Programs

Programs 2 and 3 are adaptations of one of my data plotting programs. Programs 1 and 2 are for the 8K expander, while program 3 is for the unexpanded VIC (or 3K expander). In each case, lines 6000–6180 comprise a subroutine that can be added to your programs. To use them, load your data into an integer array, A%. Put your title into a string variable, N$. A GOSUB 6000 will now display your data as a graph. Press any key to return to the main program. The routine automatically scales the graph to fit the data and prints the high and low limits on the screen. Note that only integer data in the range from −32768 to +32767 may be used. Remember, with an 8K expander, Program 1 must be loaded and run first.

I have made extensive use of multiple statements per line in addition to using the integer array in order to save memory. In an unexpanded VIC, Program 3 will leave about 1.6K for your main program.

Program 1. 8K Setup

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 POKE44,26:POKE6656,0                                  :rem 40
20 POKE648,24:POKE36869,224                               :rem 2
50 PRINT"{CLR}":NEW                                     :rem 237
```

Program 2. Graph Expanded

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 DIMA%(95):FORI=8TO511:POKE4096+I,PEEK(32768+I):NEXT
                                                        :rem 194
20 FORI=4096TO4103:READJ:POKEI,J:NEXT:DATA0,0,0,8,8,0,0,0
                                                        :rem 102
30 FORI=0TO95:A%(I)=SIN(I/12)*100:NEXT                  :rem 210
40 N$="PLOT OF SIN FUNCTION"                            :rem 123
50 GOSUB6000                                            :rem 171
60 END                                                   :rem 61
6000 HI=-32768:LO=32767:FORI=0TO95                      :rem 206
6010 IFHI<A%(I)THENHI=A%(I)                              :rem 32
6020 IFLO>A%(I)THENLO=A%(I)                              :rem 55
6030 NEXT:DF=HI-LO                                       :rem 98
6040 POKE36869,236:POKE36879,25                         :rem 221
6050 PRINT"{CLR}"                                        :rem 47
6060 FORI=37888TO38393:POKEI,6:NEXT                     :rem 133
6070 FORI=4608TO6143:POKEI,0:NEXT                         :rem 4
6080 FORI=0TO11:FORJ=0TO15:POKE6173+22*J+I,64+16*I+J:NEXT:NEX
     T                                                    :rem 4
6090 FORI=0TO13:POKE6150+I,0:POKE6524+I,0:NEXT          :rem 251
6100 FORI=0TO15:POKE6172+22*I,0:POKE6185+22*I,0:NEXT     :rem 24
6120 HI$=STR$(HI):LO$=STR$(LO)                          :rem 249
6130 PRINT"{HOME}{DOWN}"TAB(6-LEN(HI$))HI$:PRINT"{14 DOWN}"TA
     B(6-LEN(LO$))LO$:PRINT"{3 DOWN}"N$                 :rem 198
6140 FORX=0TO95                                         :rem 136
6150 Y=INT(127*(HI-A%(X))/DF):BY=4608+128*INT(X/8)+Y:BI=8*INT
     (X/8)+7-X                                          :rem 210
6160 POKEBY,PEEK(BY)OR(2↑BI):NEXT                        :rem 90
6170 GETZ$:IFZ$=""THEN6170                              :rem 241
6180 POKE36869,224:PRINT"{CLR}":RETURN                  :rem 138
```

Program 3. Graph Unexpanded

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 POKE52,26:POKE56,26:CLR                               :rem 14
20 DIMA%(79)                                             :rem 99
30 FORI=0TO79:A%(I)=SIN(I/12)*100:NEXT                  :rem 212
40 N$="PLOT OF SIN FUNCTION"                            :rem 123
50 GOSUB6000                                            :rem 171
60 END                                                   :rem 61
```

```
6000 HI=-32768:LO=32767:FORI=0TO79                          :rem 208
6010 IFHI<A%(I)THENHI=A%(I)                                  :rem 32
6020 IFLO>A%(I)THENLO=A%(I)                                  :rem 55
6030 NEXT:DF=HI-LO                                           :rem 98
6040 POKE36869,239:POKE648,26:POKE36879,25:PRINT"{CLR}"
                                                             :rem 29
6050 FORI=6656TO7161:POKEI,160:NEXT                         :rem 111
6060 FORI=38400TO38905:POKEI,6:NEXT                         :rem 113
6070 FORI=7168TO8127:POKEI,0:NEXT                            :rem 12
6080 FORI=0TO9:FORJ=0TO11:POKE6685+22*J+I,12*I+J:NEXT:NEXT
                                                             :rem 70
6090 FORI=0TO9:POKE6663+I,192:POKE6949+I,192:NEXT           :rem 188
6100 FORI=0TO11:POKE6684+22*I,221:POKE6695+22*I,221:NEXT
                                                            :rem 236
6110 POKE6662,240:POKE6673,238:POKE6948,237:POKE6959,253
                                                            :rem 173
6120 HI$=STR$(HI):LO$=STR$(LO)                              :rem 249
6130 PRINT"{HOME}{RVS}{DOWN}"TAB(6-LEN(HI$))HI$:PRINT"
     {10 DOWN}{RVS}"TAB(6-LEN(LO$))LO$:PRINT"{3 DOWN}{RVS}"N$
                                                            :rem 184
6140 FORX=0TO79                                             :rem 138
6150 Y=INT(95*(HI-A%(X))/DF):BY=7168+96*INT(X/8)+Y:BI=8*INT(X
     /8)+7-X                                                :rem 126
6160 POKEBY,PEEK(BY)OR(2↑BI):NEXT                            :rem 90
6170 GETZ$:IFZ$=""THEN6170                                  :rem 241
6180 POKE648,30:POKE36869,240:PRINT"{CLR}":RETURN            :rem 34
```

Daniel Gray

Sprite-Imation

Sprites are high-resolution, smoothly moving characters often used in graphic displays and game programs. Formerly, sprites were available only on computers equipped with special hardware, but now the VIC can also create sprites for a wide variety of uses. People familiar with sprite programming on the Commodore 64 should have no problem adapting sprites for use on the VIC.

Sprites

First, let's get the machine language program necessary to create the sprites in memory so that you can experiment as you read. Once you have the machine language routine in memory, all commands can be entered through a BASIC program. But more on that later.

Turn your VIC off and attach an 8K or greater expander. (Although "Sprite-Imation" needs no expansion, the loader program does.) Now type in and save Program 1 using the filename SPRITE LOADER. Program 1 is a BASIC loader program that will write the machine language program to disk using the filename SPRITE-IMATION. Tape users should change the *,8,0* in line 50 to *,1,0*. When you have saved Program 1, enter the following line in direct mode:

POKE 44,30:POKE 7680,0:NEW

Now, load Program 1 and run it. The program will take about a minute to POKE the machine language program into memory and save it out to disk or tape. If you have made an error, the program will tell you. Correct the data, resave the program, and run it again. If the program runs successfully, you will have the machine language on tape or disk under the filename SPRITE-IMATION.

Sprite-Imation works with an unexpanded VIC or with 3K expansion memory. To use it, load the machine language:

LOAD "SPRITE-IMATION",8,1 (use ,1,1 for tape)

Once it is loaded, enter

SYS 6000

to activate the machine language. You'll see a new startup screen with Sprite-Imation printed across the top. Now, type in Program 2, the demo program, save it, and then run it.

The remainder of this article will explain what you'll need to know to use this routine to create your own sprite.

What Shape? What Color? Where?

All you have to do is tell Sprite-Imation what shape to be, what color to be, and where to be drawn on the screen. Sprites may be any of 8 colors in standard mode or any of 16 colors in multicolor mode. Control of the sprites may be accomplished via special BASIC commands, POKE statements, or 6502 machine language. Four sprites are available on the VIC, and these are larger than the standard-sized sprites available on other Commodore computers.

Features of the VIC sprites

1. 16 horizontal bits × 24 vertical bits
2. Individual color control of sprites
3. Sprite multicolor mode
4. Variable sprite-to-sprite priority
5. Sprite-sprite collision detection
6. Sprite-background collision detection

These capabilities make it simple to program large, smoothly moving characters on the VIC. Using the special graphic commands available through Sprite-Imation, it is possible to produce high-quality, animated graphic displays in BASIC.

Sprite-Imation creates four sprites numbered 0–3. To create a sprite, the proper numbers must be POKEd into each of five registers, or used with the BASIC commands *GR* (the back-arrow character) to clear the screen and enter sprite mode, *SP* (the @ character) to give each sprite its position, and *D* (£ character) to draw the sprite. (Each of the three BASIC commands will be printed on the VIC screen with a special reverse-type character.)

Figure 1 is a chart of the sprite registers. Although it may appear complicated, only three registers are necessary to create a sprite in addition to its X,Y position registers.

Defining the Sprite

The sprite is defined in the same way that custom characters are defined, but more data is required since the sprite is larger. The sprite consists of six characters, therefore a total of 48 bytes are needed to define a sprite. This data may be placed in any portion of memory with some exceptions. Data cannot be placed in memory below 1020 ($03FC) or in the area occupied by the Sprite-Imation program, 5754–6144 ($167A–$1800) and 7136–7679 ($1BE0–$1DFF). The remaining memory of 2649 bytes may be used to store data or programs. Even more memory is available with 3K expansion memory.

The most efficient place for data is within the redefined character matrix, 6144–6656 ($1800–$1A00). However, care must be taken here since the user should not place data in the same memory occupied by the alphabetic or numeric sequence. A complete memory map of Sprite-Imation is shown in figure 2.

Figure 1. Important Sprite-Imation Registers

	Sprite Number			
Function	**0**	**1**	**2**	**3**
Location of sprite data	720 $02D0	721 $02D1	722 $02D2	723 $02D3
Position of sprite on screen (X)	730 $02DA	732 $02DC	734 $02DE	736 $02E0
Position of sprite on screen (Y)	731 $02DB	733 $02DD	735 $02DF	737 $02E1
Enable sprite (on if <> 0)	740 $02E4	741 $02E5	742 $02E6	743 $02E7
Sprite multicolor mode (on if <> 0)	746 $02EA	746 $02EA	746 $02EA	746 $02EA
Sprite foreground color (0–7)	750 $02EE	751 $02EF	752 $02F0	753 $02F1
Sprite auxiliary color (0–255)	36878 $900E	36878 $900E	36878 $900E	36878 $900E
Sprite-sprite collision (0=none)	756 $02F4	757 $02F5	758 $02F6	759 $02F7
Sprite-background collision (0=none)	762 $02FA	763 $02FB	764 $02FC	765 $02FD
Number of sprites (1–4)	725 $02D5	725 $02D5	725 $02D5	725 $02D5
Alternate joystick register (X: 0–2)	726 $02D6	726 $02D6	726 $02D6	726 $02D6
Alternate joystick register (Y: 0–2)	727 $02D7	727 $02D7	727 $02D7	727 $02D7
Alternate joystick register (fire button)	728 $02D8	728 $02D8	728 $02D8	728 $02D8

One method of placing sprite data within the character matrix is to first POKE 36869,240. This moves the character matrix from 6144 to its default value at 32768. However, if this is done, you must POKE 36869,254 before sprites are placed on the screen. Also, characters cannot be printed or POKEd onto the same screen as sprites are located with this method. This method does let you use as many as four sprites and 40 custom characters simultaneously, however. While the Sprite-Imation program is running with 254 in

36869, only the first 64 characters of the character matrix may be used. All reverse characters and graphic characters are unavailable unless register 36869 is returned to its default value of 240.

Another place to locate sprite data is below the Sprite-Imation program. To do this, you must lower the top of memory below the sprite data.

Figure 2. Large-Scale Memory Map of Sprite-Imation

Function	Start Address	End Address
Sprite-Imation registers and data	720 $02D0	765 $02FD
Vector addresses and system variables for sprites	766 $02FE	1020 $03FC
3K Expansion RAM memory	1024 $0400	4095 $0FFF
User BASIC memory in RAM	4096 $1000	5753 $1679
Sprite-Imation interpreter and subroutines	5754 $167A	6143 $17FF
Sprite-Imation character matrix in RAM	6144 $1800	6655 $19FF
Sprite-Imation work area and main program	6656 $1A00	7679 $1DFF

Sprite data is calculated in the same way that data is calculated for custom characters. The difference between VIC-20 sprites and Commodore 64 sprites is that VIC sprite data is calculated vertically instead of horizontally as

Byte 0	Byte 24
Byte 1	Byte 25
Byte 2	Byte 26
.	.
.	.
.	.
Byte 23	Byte 47

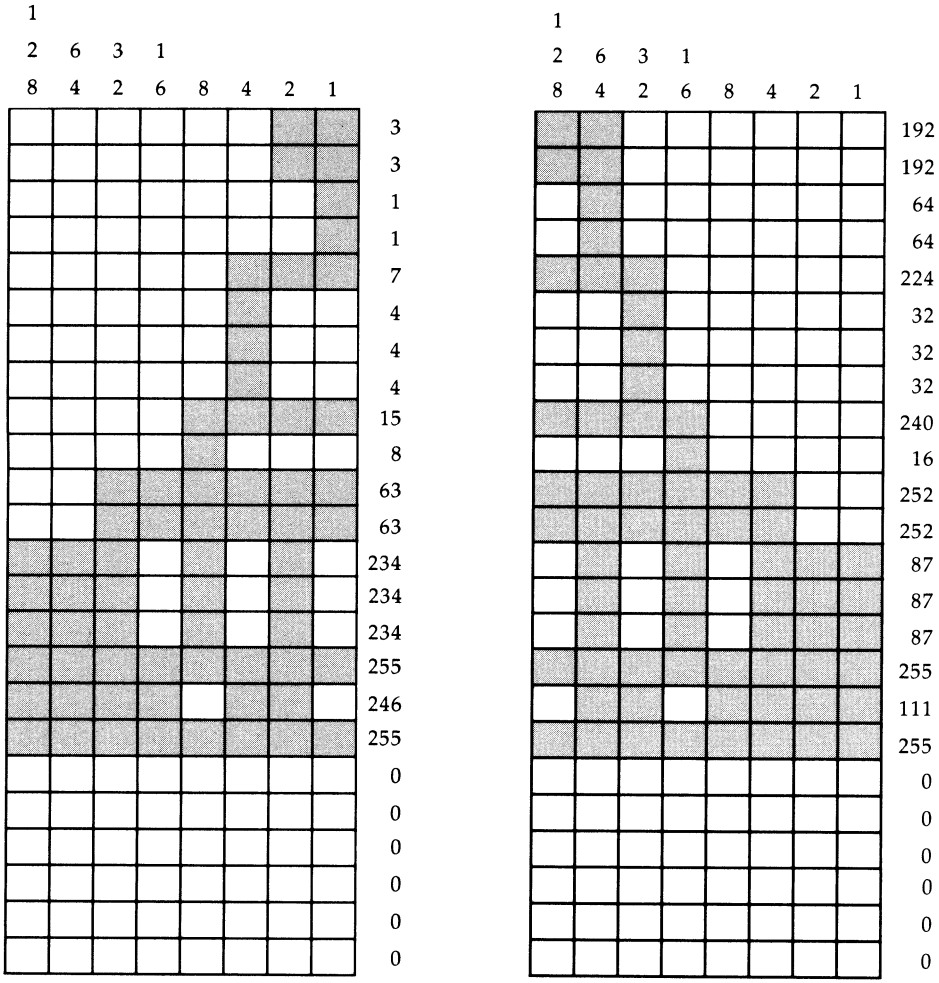
To calculate the sprite data, you must first draw your sprite on a grid of 16 dots × 24 dots. Graph paper is the easiest thing to use for this purpose. Outline an area corresponding to the 16 block × 24 block size of the sprite. Fill in the blocks in the pattern of your sprite. The blocks that are filled in will appear to be "on" when the sprite is projected onto the screen. Any

background characters placed on the screen will appear to be behind your sprite.

The computer, however, cannot tell what a sprite looks like until the shape of the sprite is given to it in the form of data. The computer requires numbers that it can understand, and you must calculate these numbers for the computer.

Using two 8 × 24 grids you can calculate the values; figure 3 is an illustration. Just fill in the boxes you need. Then, using the values at the top of each column, add the values of the filled-in boxes for each row.

Figure 3. Sprite Grid Illustration



This data can now be POKEd into memory but Sprite-Imation must also be told where in memory to find sprite information. This is done by POKEing the proper number into registers 720–723 ($02D0–$02D3). Memory is divided into 48-byte blocks for this purpose, and the number *N* used in registers 720–723 may be multiplied by 48 to find the location of the sprite data.

Demonstration

A simple demonstration program is given below. The REM statements in the program may be deleted since they serve only to clarify the program and do not affect its function.

```
10 POKE56,22:POKE52,22:POKE55,32:POKE51,32:REM LOWER TOP OF M
   EMORY
20 FORI=5664TO5711:READA:POKEI,A:NEXT:REM POKE SPRITE DATA IN
   TO MEMORY
30 DATA3,3,1,1,7,4,4,4,15,8,63,63,234,234,234,255,246,255,0,0
   ,0,0,0,0
40 DATA192,192,64,64,224,32,32,32,240,16,252,252,87,87,87,255
   ,111,255
50 DATA0,0,0,0,0,0
60 POKE720,118:REM TELL SPRITE 0 WHERE TO FIND DATA
70 POKE750,7:REM MAKE SPRITE 0 YELLOW
80 POKE36879,106:REM MAKE SCREEN BLUE
90 POKE725,1:REM MAKE ONLY FIRST SPRITE
100 LET ←:REM SPRITE MODE
110 FORI=0TO220:LET @ 0,I,I:REM POSITION SPRITE 0
120 LET £:NEXT:REM DRAW SPRITE
130 GOTO110
```

The sprite can be drawn in multicolor mode with the addition of the following statements:

```
94 POKE746,1:REM PUT IN MULTICOLOR MODE
96 POKE36878,255:REM GIVE AUXILIARY COLOR
```

Multicolor sprites are defined in terms of "bit pairs" instead of bits. Each bit pair can represent one of four colors:

00 = Background color
01 = Border color
10 = Foreground color
11 = Auxiliary color

Sprite Positioning

Once the sprite is defined, it must be placed on the screen. You can do this in two ways: You can POKE the X,Y coordinates of the sprite into the position registers shown in figure 1, or you can use the *SP* statement (@ key) with X,Y coordinates. The proper syntax for this statement is

LET *SP* N,X,Y

If this statement is used at the end of a line, it must be followed by a semicolon. Multiple Sprite-Imation statements may follow a single LET statement:

LET *SP* 0,X,Y:*SP* 1,X,Y:*SP* 2,X,Y:*SP* 3,X,Y:*D*

where the *D* statement (£ key) implements the Sprite-Imation program. Any X or Y between 32 and 208 will appear on the screen.

Turning a Sprite Off

A sprite is turned on by placing a one in its enable register shown in figure 1 or by using the Sprite-Imation *SP* statement. Once turned on, it will remain on until a zero is POKEd into its enable register. Also, a sprite may be temporarily turned off by placing a lower number in register 725 ($02D5). For instance, a two placed in 725 will cause the Sprite-Imation program to draw only sprites 0 and 1. If a one is placed in 725, it would produce only sprite 0. The fewer sprites used in this manner, the faster the sprites may be moved around on the screen.

Program 1. Sprite Loader

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}PLEASE WAIT..."                          :rem 62
20 FORI=5754TO6658:READA:POKEI,A:C=C+A:NEXT           :rem 184
30 FORI=7136TO7679:READA:POKEI,A:C=C+A:NEXT           :rem 185
40 IFC<>131606THENPRINT"TYPING ERROR IN DATA":STOP    :rem 119
50 SYS57809"0:SPRITE-IMATION",8,0:REM USE ,1,0 FOR TAPE
                                                      :rem 201
60 POKE193,122:POKE194,22:POKE174,0:POKE175,30:SYS63109
                                                      :rem 188
100 DATA 13,13,156,29,29,29,29,83,80,82,73             :rem 26
110 DATA 84,69,45,73,77,65,84,73,79,78,31               :rem 4
120 DATA 13,0,201,136,240,10,201,58,176,3,76           :rem 91
130 DATA 128,0,76,138,0,32,79,23,32,115,0             :rem 208
140 DATA 201,0,208,5,32,93,23,16,231,201,92            :rem 42
150 DATA 208,19,173,213,2,208,2,169,4,201,5            :rem 55
160 DATA 176,250,141,238,29,32,220,29,144,221,201      :rem 97
170 DATA 95,208,7,169,147,32,210,255,144,210,201       :rem 56
180 DATA 58,208,3,56,176,203,201,64,208,186,32        :rem 222
```

```
190 DATA 115,0,176,45,201,52,176,41,56,233,47          :rem 165
200 DATA 168,153,227,2,10,133,158,32,115,0,160         :rem 194
210 DATA 2,132,159,32,253,206,32,138,205,32,247        :rem 253
220 DATA 215,152,164,158,153,216,2,230,158,164,159     :rem 160
230 DATA 136,208,232,240,149,76,8,207,32,80,23         :rem 214
240 DATA 78,34,145,173,32,145,9,127,106,45,17          :rem 173
250 DATA 145,170,10,10,10,138,105,0,41,95,73            :rem 93
260 DATA 126,201,96,176,2,73,96,160,3,170,41           :rem 117
270 DATA 3,201,2,176,2,73,1,153,213,2,138              :rem 203
280 DATA 74,74,136,208,238,56,46,34,145,32,90          :rem 183
290 DATA 23,76,112,247,120,141,161,2,142,162,2         :rem 197
300 DATA 140,163,2,96,173,161,2,174,162,2,172          :rem 155
310 DATA 163,2,88,96,189,60,3,133,158,189,61           :rem 138
320 DATA 3,133,159,96,120,24,162,122,160,22,32         :rem 198
330 DATA 153,255,32,164,227,162,4,189,165,23,149        :rem 70
340 DATA 123,202,208,248,169,254,141,5,144,169,17      :rem 114
350 DATA 160,23,141,40,3,140,41,3,166,51,32             :rem 40
360 DATA 4,228,169,122,160,22,32,30,203,88,76          :rem 165
370 DATA 129,227,234,76,146,22,234,169,30,133,147      :rem 119
380 DATA 165,191,240,59,224,26,176,55,192,27,176        :rem 83
390 DATA 51,138,56,233,4,48,45,170,152,56,233          :rem 175
400 DATA 4,48,38,168,10,133,195,10,133,146,10          :rem 155
410 DATA 10,24,101,195,24,101,146,134,146,24,101        :rem 29
420 DATA 146,192,11,144,8,208,4,224,14,144,2           :rem 104
430 DATA 230,147,133,146,133,91,96,234,169,32,133      :rem 113
440 DATA 147,76,231,23,96,26,192,26,32,27,128          :rem 178
450 DATA 27,128,27,128,27,145,181,181,145,215,215      :rem 116
460 DATA 151,255,24,36,66,126,66,66,66,0,124           :rem 126
470 DATA 34,34,60,34,34,124,0,28,34,64,64              :rem 219
480 DATA 64,34,28,0,120,36,34,34,34,36,120               :rem 6
490 DATA 0,126,64,64,120,64,64,126,0,126,64             :rem 63
500 DATA 64,120,64,64,64,0,28,34,64,78,66              :rem 232
510 DATA 34,28,0,66,66,66,126,66,66,66,0               :rem 186
520 DATA 28,8,8,8,8,8,28,0,14,4,4                       :rem 82
530 DATA 4,4,68,56,0,66,68,72,112,72,68                :rem 137
540 DATA 66,0,64,64,64,64,64,64,126,0,66               :rem 184
550 DATA 102,90,90,66,66,66,0,66,98,82,74              :rem 247
560 DATA 70,66,66,0,24,36,66,66,66,36,24               :rem 190
570 DATA 0,124,66,66,124,64,64,64,0,24,36              :rem 225
580 DATA 66,66,74,36,26,0,124,66,66,124,72              :rem 34
590 DATA 68,66,0,60,66,64,60,2,66,60,0                  :rem 80
600 DATA 62,8,8,8,8,8,8,0,66,66,66                     :rem 148
610 DATA 66,66,66,60,0,66,66,66,36,36,24               :rem 191
620 DATA 24,0,66,66,66,90,90,102,66,0,66               :rem 177
630 DATA 66,36,24,36,66,66,0,34,34,34,28               :rem 182
640 DATA 8,8,8,0,126,2,4,24,32,64,126                   :rem 17
650 DATA 0,60,32,32,32,32,32,60,0,135,219              :rem 199
660 DATA 221,213,221,219,135,255,60,4,4,4,4             :rem 53
670 DATA 4,60,0,0,8,28,42,8,8,8,8                       :rem 81
680 DATA 135,183,189,164,181,181,133,255,0,0,0         :rem 217
690 DATA 0,0,0,0,0,8,8,8,8,0,0                         :rem 161
```

```
700 DATA 8,0,36,36,36,0,0,0,0,0,36                    :rem 101
710 DATA 36,126,36,126,36,36,0,8,30,40,28             :rem 217
720 DATA 10,60,8,0,0,98,100,8,16,38,70                 :rem 61
730 DATA 0,48,72,72,48,74,68,58,0,4,8                  :rem 40
740 DATA 16,0,0,0,0,0,4,8,16,16,16                    :rem 101
750 DATA 8,4,0,32,16,8,8,8,16,32,0                    :rem 122
760 DATA 8,42,28,62,28,42,8,0,0,8,8                   :rem 183
770 DATA 62,8,8,0,0,0,0,0,0,0,8                       :rem 208
780 DATA 8,16,0,0,0,126,0,0,0,0,0                      :rem 41
790 DATA 0,0,0,0,24,24,0,0,2,4,8                      :rem 252
800 DATA 16,32,64,0,60,66,70,90,98,66,60              :rem 181
810 DATA 0,8,24,40,8,8,8,62,0,60,66                   :rem 175
820 DATA 2,12,48,64,126,0,60,66,2,28,2                 :rem 64
830 DATA 66,60,0,4,12,20,36,126,4,4,0                   :rem 2
840 DATA 126,64,120,4,2,68,56,0,28,32,64              :rem 172
850 DATA 124,66,66,60,0,126,66,4,8,16,16              :rem 179
860 DATA 16,0,60,66,66,60,66,66,60,0,60               :rem 127
870 DATA 66,66,62,2,4,56,0,0,0,8,0                    :rem 122
880 DATA 0,8,0,0,0,0,8,0,0,8,8                        :rem 162
890 DATA 16,14,24,48,96,48,24,14,0,0,0                 :rem 71
900 DATA 126,0,126,0,0,0,112,24,12,6,12                :rem 83
910 DATA 24,112,0,60,66,2,12,16,0,16,0                 :rem 43
920 DATA 96,240,248                                   :rem 176
930 DATA 169,0,133,147,133,146,153,244,2,153,250       :rem 56
940 DATA 2,185,208,2,24,121,208,2,38,147,24            :rem 62
950 DATA 121,208,2,38,147,162,4,10,38,147,202         :rem 160
960 DATA 208,250,133,146,152,10,141,255,2,168,185     :rem 110
970 DATA 244,23,190,245,23,133,176,134,177,169,26     :rem 127
980 DATA 133,88,133,90,190,218,2,185,219,2,134        :rem 229
990 DATA 196,160,2,72,41,7,192,2,240,7,174             :rem 25
1000 DATA 234,2,240,2,41,254,153,201,2,104,74         :rem 132
1010 DATA 74,74,153,199,2,165,196,136,208,227,173     :rem 126
1020 DATA 203,2,133,89,24,105,8,133,87,160,95         :rem 163
1030 DATA 169,0,153,0,26,136,16,250,24,160,24         :rem 146
1040 DATA 136,177,146,145,89,152,72,105,24,168,177    :rem 177
1050 DATA 146,145,87,104,168,208,238,160,24,166,89    :rem 178
1060 DATA 132,251,172,202,2,240,13,24,126,0,26        :rem 184
1070 DATA 126,32,26,126,64,26,136,208,243,232,164     :rem 106
1080 DATA 251,136,208,230,162,0,134,89,160,0,142       :rem 44
1090 DATA 198,2,140,199,2,138,10,10,24,109,199        :rem 219
1100 DATA 2,133,87,32,243,29,101,87,133,251,10        :rem 198
1110 DATA 170,32,101,23,173,200,2,24,109,198,2        :rem 188
1120 DATA 170,173,201,2,109,199,2,168,32,170,23       :rem 254
1130 DATA 160,0,166,251,165,146,197,158,208,6,165     :rem 117
1140 DATA 147,197,159,240,30,177,158,56,229,196,201   :rem 229
1150 DATA 76,144,21,201,88,176,17,189,180,3,145        :rem 19
1160 DATA 158,165,159,24,105,120,133,159,189,0,148    :rem 167
1170 DATA 145,158,177,146,133,195,72,165,147,24,105   :rem 224
1180 DATA 120,133,92,177,91,168,104,201,76,144,19     :rem 113
1190 DATA 176,8,255,255,255,255,255,255,255,255,233   :rem 235
1200 DATA 76,170,189,180,3,188,0,148,201,32,240         :rem 4
```

```
1210 DATA 6,174,252,3,157,250,2,166,251,157,180          :rem 5
1220 DATA 3,152,157,0,148,160,0,177,146,56,233          :rem 208
1230 DATA 76,197,251,144,20,189,180,3,133,195,174       :rem 121
1240 DATA 252,3,189,238,2,204,234,2,240,2,9              :rem 56
1250 DATA 8,145,91,169,23,141,114,29,165,195,201         :rem 64
1260 DATA 76,144,13,201,88,176,3,141,244,2,174          :rem 215
1270 DATA 252,3,157,244,2,56,238,114,29,233,32          :rem 215
1280 DATA 48,4,133,195,16,244,184,165,195,10,10          :rem 14
1290 DATA 10,141,113,29,185,0,25,17,89,145,89           :rem 174
1300 DATA 200,192,8,208,244,160,0,165,87,24,105         :rem 254
1310 DATA 64,145,146,165,251,10,170,165,146,157,60      :rem 157
1320 DATA 3,165,147,157,61,3,165,89,24,105,8            :rem 123
1330 DATA 133,89,174,198,2,172,199,2,200,192,4          :rem 226
1340 DATA 240,3,76,143,28,232,224,3,240,3,76            :rem 106
1350 DATA 141,28,160,95,185,0,26,145,176,136,16          :rem 15
1360 DATA 248,32,243,29,10,170,169,76,24,101,196         :rem 65
1370 DATA 168,105,12,133,195,152,72,32,101,23,160        :rem 95
1380 DATA 0,104,145,158,168,200,232,232,196,195,208     :rem 211
1390 DATA 238,96,160,0,140,252,3,185,228,2,133          :rem 214
1400 DATA 191,32,224,27,172,252,3,200,192,4,208         :rem 250
1410 DATA 237,24,96,173,255,2,10,133,196,10,24          :rem 209
1420 DATA 101,196,133,196,96                            :rem 105
```

Program 2. Sprite Demo

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 POKE56,22:POKE52,22:POKE55,32:POKE51,32               :rem 131
10 FORI=5664TO5711:READA:POKEI,A:NEXT                    :rem 67
20 DATA0,0,3,15,31,55,113,112,248,248,252,223,255,255,254,254
   ,254,127,127,62,29,15,3,0                            :rem 165
30 DATA0,0,192,240,248,236,158,142,15,31,31,111,191,15,7,7,7,
   14,30,124,248,240,192,0                               :rem 55
40 POKE720,118:POKE750,6:POKE725,1:POKE36879,14:LET←::rem 128
45 FORI=0TO50:A=INT(RND(1)*505):POKE7680+A,46:POKE38400+A,INT
   (RND(1)*6)+1:NEXT                                     :rem 67
48 POKE7910,42:POKE38630,7                              :rem 206
50 FORI=0TO6.28STEP.06:X=060*COS(I)+107:Y=050*SIN(I)+107:LET@
   0,X,Y:£:NEXT:GOTO50                                  :rem 105
```

Thomas
Gronos

VIC Draw McGraw

"VIC Draw McGraw" can be used to easily create graphic pictures on your 8K expanded VIC. With it, you can quickly design characters and figures to suit almost any need. You can even use it to draw hi-res pictures on the screen.

Once you've typed in and saved VIC Draw McGraw, make sure you have an 8K expander cartridge plugged into the port at the back of your computer. Load the program, run it, and you'll see a 14-item menu screen:

1 Move Cursor
2 Print Character
3 Rotate Cursor Character
4 Change Character Size
5 Image to Printer
6 Use Programmable Characters
7 Condense Characters
8 Save to Printer
9 Save to Tape
10 Restore from Tape
11 Draw with Joystick
12 Erase Mode
13 Print Mode
14 Color of the Screen

Beneath this menu is additional information on the size of the character grid you're presently working with, the type of character, the number of the character under the cursor, the mode VIC Draw McGraw is in, and the number of characters remaining to you. (All these will be explained shortly.)

How It All Works

Move Cursor. Function 1 from the main menu is probably the most widely called function of VIC Draw McGraw. As soon as you hit the 1 key and press RETURN (all functions require that you follow the number with the RETURN keypress), the screen clears and you'll see a cursor at the top-left corner of the screen. You can move it anywhere simply by using the normal cursor keys. This function is used to position the cursor in the desired place, something important for many of the other functions. Hit the RETURN key again to go back to the main menu.

Print Character. When you press the 2 key and hit RETURN, you'll see a prompt at the bottom of the menu screen. Enter the screen code (often called the POKE code) for the character you want to print at the present cursor position on the alternate screen. You'll also be asked if you want the

reverse mode (answer Y for Yes, N for No), the set number (1 for uppercase/graphics, 2 for lowercase/uppercase), and whether you want the pixels of the character shifted to the right and down (answer N for No, Y for Yes; if the latter, you'll have to provide values for both the right and down SHIFTs). When you shift a character's pixels, it still prints, but it won't necessarily appear directly beneath the cursor position. You can use this feature to print "offset" characters almost anywhere on the screen.

Rotate Cursor Character. Hitting the 3 key calls this function. Any character under the cursor (remember, you can move the cursor by pressing the 1 key) can be rotated 90, 180, or 270 degrees. You can make your characters, whether standard or those you designed, lie on their sides or even stand on their heads.

Change Character Size. By pressing the 4 key, you can change the character size. (This is also used to set the drawing area when you enter the Draw with Joystick mode.) All that happens is that after entering the width and height values, the Size message in the middle box of the main menu changes. To actually draw enlarged characters, however, you'll have to go back to the alternate screen and print a character. Try it out.

Image to Printer. Pressing the 5 key, assuming you have a printer connected to your VIC, sends the screen graphics to the printer. The process is quite long and takes approximately 55 minutes to run. The results, however, are worth the wait.

Use Programmable Characters. Hitting the 6 key toggles between the VIC's standard character set and the programmable characters you've already created. You can create your own custom characters by accessing the Draw with Joystick mode, drawing a character, and then noting the character number under the cursor (that information is listed in the middle message box on the main menu). When you're in the programmable character mode, if you try to print a character on the alternate screen, you'll see only two messages: Screen Code and Shift. Instead of entering a screen code value, type in the character number (the value you see in the message box when the cursor is directly over your custom character).

Condense Characters. This function, called by pressing the 7 key, replaces programmable characters that are all clear or all filled with the predefined characters used for the cursor and the background. Use this function with caution and only if you're running out of characters (256 possible characters) or space on the screen (256 possible locations). Though it sometimes may appear as if nothing but character shifting is happening, the function does free up some screen space when you're cramped.

Save to Printer and **Save to Tape.** Pressing either the 8 or 9 key saves the character data you've created, sending it either to your printer (assuming one is connected) or to your tape drive. When you send the data to the printer, you have to specify whether only the character under the cursor is

recorded or whether the presently selected size is used and all character data is printed. The numbers shown on a printout (as well as those sent to tape if you select 9) indicate the bit values of each byte. If you send the data to tape, you have to provide a filename.

Restore from Tape. This feature, accessed by entering 10, loads your character data from tape back into the computer. Make sure you provide the correct filename.

Draw with Joystick. This is one of the most interesting features of VIC Draw McGraw. After setting the character size (by using function 4), you can enter 11 and hit RETURN. The alternate screen appears, and you can use a joystick to draw on the hi-res screen. Pressing the P, E, or C key while drawing changes the function to set Print, Erase, or Change modes. Press the joystick button to draw, moving the joystick around the screen. If you choose Erase, when you press the button, any point you move over is erased; if you choose Change, then printed points are erased and erased points are printed. Note that the drawing area is established by the Change Character Size function.

Erase Mode. When you enter 12, having earlier placed the cursor over the desired character, you can erase portions of the character by printing over it with another character. Use this to clear selected characters from the alternate screen.

Print Mode. Enter 13 to go back to printing mode.

Color of the Screen. This allows you to select character colors for any character you wish. Simply follow the prompts which appear in the bottom of the main menu screen.

VIC Draw McGraw is a powerful program. With it, you can create almost any custom characters you want. You can even enlarge the VIC's standard character set and print that. Experiment with the program—you'll probably find uses for it I never thought of!

VIC Draw McGraw

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

Some lines of this program require keywords to be abbreviated so that they will not exceed the four-screen-line limit. See Appendix B.

```
1 REM***VICDRAWMCGRAW ****************************************
  **********XXXXXXXX                                    :rem 46
2 REM*****************************************************
  **********XXXXXXXX                                   :rem 170
3 REM*****************************************************
  **********XXXXXXXX                                   :rem 171
4 REM*****************************************************
  **********XXXXXXXX                                   :rem 172
```

```
5 REM*****************************************************
  **********XXXXXXXX                                :rem 173
6 REM*****************************************************
  **********XXXXXXXX                                :rem 174
7 REM*****************************************************
  **********XXXXXXXX                                :rem 175
8 REM*****************************************************
  **********XXXXXXXX                                :rem 176
9 REM*****************************************************
  **********XXXXXXXX                                :rem 177
10 REM****************************************************
   ***********XXXXXXXX                                :rem 217
11 REM****************************************************
   ***********XXXXXXXX                                :rem 218
12 REM****************************************************
   ***********XXXXXXXX                                :rem 219
13 REM****************************************************
   ***********XXXXXXXX                                :rem 220
14 REM****************************************************
   ***********XXXXXXXX                                :rem 221
15 REM****************************************************
   ***********XXXXXXXX                                :rem 222
16 REM****************************************************
   ***********XXXXXXXX                                :rem 223
17 REM****************************************************
   ***********XXXXXXXX                                :rem 224
18 REM****************************************************
   ***********XXXXXXXX                                :rem 225
19 REM****************************************************
   ***********XXXXXXXX                                :rem 226
20 REM****************************************************
   ***********XXXXXXXX                                :rem 218
21 REM****************************************************
   ***********XXXXXXXX                                :rem 219
22 REM****************************************************
   ***********XXXXXXXX                                :rem 220
23 REM****************************************************
   ***********XXXXXXXX                                :rem 221
24 REM****************************************************
   ***********XXXXXXXX                                :rem 222
25 REM****************************************************
   ***********XXXXXXXX                                :rem 223
26 REM****************************************************
   ***********XXXXXXXX                                :rem 224
27 REM****************************************************
   ***********XXXXXXXX                                :rem 225
28 REM****************************************************
   ***********XXXXXXXX                                :rem 226
29 REM****************************************************
   ***********XXXXXXXX                                :rem 227
```

```
30 REM*****************************************************
   ***********XXXXXXXX                                :rem 219
31 REM*****************************************************
   ***********XXXXXXXX                                :rem 220
32 REM*****************************************************
   ***********XXXXXXXX                                :rem 221
33 IFPEEK(833)=31THENPOKE833,32:GOTO46                 :rem 20
42 POKE36879,92:PRINT"{CLR}";TAB(158);"{BLK}VICDRAWMCGRAW":PR
   INTTAB(27);"{5 DOWN}PLEASE WAIT"                    :rem 215
43 POKE5114,2:FORX=16301TO16379:READA:POKEX,A:NEXT:POKE55,173
   :POKE56,63                                           :rem 94
44 POKE44,28:FORX=5120TO7167:POKEX,0:NEXT:FORX=5128TO5135:POK
   EX,255:NEXT:DIMS(7)                                   :rem 9
45 GOSUB96:PRINT"{CLR}{BLU}";:FORX=1TO505:PRINT"@";:NEXT:POKE
   5113,0:POKE38905,6:GOSUB96                          :rem 255
46 MD$="PRINT":DM$="P":CH$="STAND":CL=254:A=4608:D=PEEK(A):PO
   KEA,1:K=0:J=1:WI=1:TL=1                             :rem 233
47 C=PEEK(5114):HC$="PROGRAM.":POKE649,1:GOTO70         :rem 31
48 PRINT"{CLR}{BLK}1{2 SPACES}MOVE CURSOR":PRINT"{WHT}2
   {2 SPACES}PRINT CHARACTER":PRINT"{BLK}3{2 SPACES}ROTATE CU
   RSOR CHAR"                                          :rem 155
51 PRINT"{WHT}4{2 SPACES}CHANGE CHAR. SIZE":PRINT"{BLK}5
   {2 SPACES}IMAGE TO PRINTER"                         :rem 151
52 PRINT"{WHT}6{2 SPACES}USE ";HC$;" CHAR."             :rem 21
54 PRINT"{BLK}7{2 SPACES}CONDENSE CHARACTERS";:PRINT"{WHT}8
   {2 SPACES}SAVE TO PRINTER":PRINT"{BLK}9{2 SPACES}SAVE TO T
   APE"                                                :rem 121
55 PRINT"{WHT}10 RESTORE FROM TAPE"                     :rem 35
56 PRINT"{BLK}11 DRAW WITH JOYSTICK":PRINT"{WHT}12 ERASE MODE
   ":PRINT"{BLK}13 PRINT MODE"                         :rem 193
57 PRINT"{WHT}14 COLOR THE SCREEN":PRINT"{PUR}{RVS}
   {22 SPACES}{BLK}";                                   :rem 75
58 WI$=MID$(STR$(WI),2,2):IFWI<10THENWI$="0"+WI$        :rem 32
59 TL$=MID$(STR$(TL),2,2):IFTL<10THENTL$="0"+TL$        :rem 33
60 PRINT"{OFF}{BLK}SIZE=";WI$;"X";TL$;"{2 SPACES}MODE=";MD$;
                                                       :rem 178
62 CL$=MID$(STR$(CL),2,3)                               :rem 97
63 PRINT"CHAR=";CH$;"{2 SPACES}# LEFT=";CL$;:PRINT"{HOME}";TA
   B(200);TAB(176);"CHAR UNDER CURSOR=";                :rem 55
64 PRINTMID$(STR$(D),2,3)                              :rem 181
66 PRINT"{RVS}{PUR}{22 SPACES}{OFF}{BLK}";:RETURN       :rem 98
70 GOSUB48:PRINT"FUNCTION? ";:GOSUB75:FU=VAL(A$):IF(FU<1)OR(F
   U>255)THEN70                                         :rem 46
73 ONFUGOSUB100,200,300,400,500,600,700,800,900,1000,1100,120
   0,1300,1400                                         :rem 106
74 ONSDGOSUB96:GOTO70                                  :rem 104
75 A$=""                                                :rem 82
76 GETB$:IFB$=""THEN76                                 :rem 255
77 IFB$=CHR$(13)THENPRINT" ":RETURN                    :rem 112
78 IFB$=CHR$(20)THENA$="DEL":PRINT" ":RETURN           :rem 100
```

```
79 IFB$<"Ø"ORB$>"9"THEN76                                  :rem 115
8Ø A$=A$+B$:PRINTB$;:IFLEN(A$)>2THENPRINT" ":RETURN         :rem 8Ø
81 GOTO76                                                   :rem 15
83 FORX=1TOTL-(SD%>Ø):FORY=1TOWI-(SR%>Ø):LU=PEEK(A+Y+22*X-23)
                                                           :rem 218
84 IFLU<2ANDCL=ØTHENONSDGOSUB96:GOSUB48:PRINT"CHARACTERS EXHA
   USTED":PRINT"USE CONDENSE":A$="R":RETURN                :rem 186
85 IFLU=1THENFORX1=ØTO7:POKE512Ø+8*C+X1,255:NEXT           :rem 231
86 IFLU<2THENPOKEA+Y+22*X-23,C:C=C+1                       :rem 154
87 CL=256-C:NEXT:NEXT:RETURN                                :rem 84
95 FORX=ØTO7:S(X)=Ø:NEXT:RETURN                             :rem 42
96 IFSD=ØTHENSD=1:POKE36866,PEEK(36866)OR128:POKE648,18:POKE3
   6879,27:POKE36869,2Ø5:RETURN                            :rem 156
97 SD=Ø:POKE36866,PEEK(36866)AND127:POKE648,16:POKE36879,92:P
   OKE36869,192:RETURN                                      :rem 16
1ØØ GOSUB96                                               :rem 128
1Ø1 GETA$:IFA$=""THEN1Ø1                                    :rem 71
1Ø5 IFA$=CHR$(13)THENGOSUB96:RETURN                       :rem 181
11Ø ON-(A$="{RIGHT}")-2*(A$="{DOWN}")-3*(A$="{LEFT}")-4*(A$="
    {UP}")GOTO115,12Ø,125,13Ø                                :rem 69
111 IF((VAL(A$)>Ø)AND(VAL(A$)<9))THENPOKEA+33792,VAL(A$)-1:GO
    TO1Ø1                                                 :rem 247
112 IF((ASC(A$)>32)AND(ASC(A$)<41))THENPOKEA+33792,ASC(A$)-25
                                                          :rem 1Ø2
113 GOTO1Ø1                                                  :rem 96
115 POKEA,D:A=A-(A<5113):GOTO135                             :rem 65
12Ø POKEA,D:A=A-22*(A<5Ø92):GOTO135                       :rem 2Ø9
125 POKEA,D:A=A+(A>46Ø8):GOTO135                             :rem 74
13Ø POKEA,D:A=A+22*(A>4629)                               :rem 2Ø3
135 D=PEEK(A):POKEA,1:GOTO1Ø1                             :rem 221
2ØØ GOSUB48:PRINT"SCREEN CODE? ";:GOSUB75:IFA$="DEL"ORA$=""TH
    EN2ØØ                                                 :rem 2Ø8
2Ø5 SC%=VAL(A$):IF(SC%<Ø)OR(SC%>255)THEN2ØØ               :rem 177
2Ø6 IF(CH$="PROG.")OR(SC%>127)THENSN%=1:GOTO215              :rem 1
2Ø7 GOSUB48:PRINT"WANT RVS ON?"                           :rem 161
2Ø8 GETA$:ON-(A$="")GOTO2Ø8:IFA$="Y"THENSC%=SC%+128       :rem 157
21Ø GOSUB48:PRINT"SET NUMBER? ";:GOSUB75:SN%=VAL(A$):IF(SN%<1
    )OR(SN%>2)THEN21Ø                                     :rem 236
215 GOSUB48:PRINT"SHIFT?"                                    :rem 76
22Ø GETA$:ON-(A$="")GOTO22Ø:IFA$<>"Y"THENSR%=Ø:SD%=Ø:GOTO239
                                                             :rem Ø
225 GOSUB48:PRINT"PIXELS TO RIGHT?"                       :rem 197
23Ø GETA$:ON-((A$<"Ø")OR(A$>"7"))GOTO23Ø:SR%=VAL(A$)       :rem 77
231 GOSUB48:PRINT"PIXELS DOWN?"                           :rem 217
232 GETA$:ON-((A$<"Ø")OR(A$>"7"))GOTO232:SD%=VAL(A$)       :rem 67
239 GOSUB96:POKEA,D                                       :rem 167
24Ø GOSUB83:IFA$="R"THENRETURN                            :rem 145
27Ø FORX=ØTO7:FORY=ØTO7                                   :rem 224
275 IF((2↑(7-Y))AND(PEEK(32768*J+512Ø*K+(SN%-1)*2Ø48+SC%*8+X)
    ))>ØTHENGOSUB283                                         :rem 1Ø
```

```
280 NEXT:NEXT:D=PEEK(A):FORX=1TO3000:NEXT:POKEA,1:RETURN
                                                    :rem 171
283 FORX1=1TOTL:FORY1=1TOWI:X2=Y*WI+SR%+Y1-1:X3=X*TL+SD%+X1-1
                                                     :rem 99
285 X4=A+INT(X2/8)+22*INT(X3/8)                    :rem 232
287 X5=PEEK(X4)*8+5120+X3-8*INT(X3/8)               :rem 53
289 IFDM$="P"THENPOKEX5,PEEK(X5)OR2↑(7-(X2-(INT(X2/8)*8))):GO
    TO299                                          :rem 183
291 POKEX5,PEEK(X5)AND(255-2↑(7-X2+(INT(X2/8)*8)))  :rem 78
299 NEXT:NEXT:RETURN                               :rem 118
300 GOSUB48:PRINT"PICK: (1) 90{2 SPACES}DEG-CW":PRINTTAB(6);"
    (2) 180 DEG-CW":PRINTTAB(6);"(3) 270 DEG-CW"    :rem 28
305 GETA$:IFA$<"1"ORA$>"3"THEN305                   :rem 62
310 GOSUB95:X=VAL(A$):GOSUB96:POKEA,D:ONXGOSUB320,350,360:FOR
    X=1TO3000:NEXT:POKEA,1:RETURN                  :rem 225
320 FORX1=7TO0STEP-1:FORX3=0TO7                    :rem 217
330 IF(PEEK(5120+D*8+X3)AND2↑X1)>0THENS(7-X1)=S(7-X1)+2↑X3
                                                   :rem 105
340 NEXT:NEXT:FORX1=0TO7:POKE5120+D*8+X1,S(X1):NEXT:RETURN
                                                   :rem 226
350 FORX4=1TO2:GOSUB320:GOSUB95:NEXT:RETURN         :rem 88
360 FORX4=1TO3:GOSUB320:GOSUB95:NEXT:RETURN         :rem 90
400 GOSUB48:PRINT"WIDTH? ";:GOSUB75:WI=VAL(A$):GOSUB48:PRINT"
    HEIGHT? ";:GOSUB75:TL=VAL(A$)                  :rem 149
410 IF(TL>23)OR(WI>22)OR(TL<1)OR(WI<1)THEN400      :rem 173
430 RETURN                                         :rem 119
500 GOSUB48:PRINT"IS PRINTER READY?":PRINT"IF SO HIT ANY KEY"
                                                   :rem 242
501 GETA$:ON-(A$="")GOTO501:GOSUB48:PRINT"...WORKING":rem 193
502 I1=4608:I2=5120:POKEA,D:OPEN4,4                :rem 244
503 Z1=0:Z2=25:Z3=6:GOSUB510:Z1=26:Z2=26:Z3=1:GOSUB510
                                                   :rem 222
505 POKEA,1:CLOSE4:RETURN                          :rem 101
510 FORX=Z1TOZ2:B$=CHR$(8)+CHR$(27)+CHR$(16)+CHR$(0)+CHR$(153
    )                                              :rem 192
515 FORX1=0TO175:CN=128:FORX2=0TOZ3                :rem 163
520 Y1=I1+INT((X*7+X2)/8)*22+INT(X1/8):Y2=I2+PEEK(Y1)*8+X*7+X
    2-8*INT((X*7+X2)/8)                            :rem 222
525 IF(PEEK(Y2)AND2↑(7-(X1-8*INT(X1/8))))THENCN=CN+2↑X2
                                                   :rem 253
530 NEXT:B$=B$+CHR$(CN):NEXT:PRINT#4,B$:NEXT:RETURN :rem 228
600 IFHC$="PROGRAM."THENHC$="STANDARD":J=0:K=1:CH$="PROG.":RE
    TURN                                            :rem 25
601 HC$="PROGRAM.":J=1:K=0:CH$="STAND":RETURN      :rem 239
700 GOSUB96:Z=0:POKEA,D:CE=0:IFC=2THENPOKEA,1:RETURN  :rem 97
705 FORX=2TOC-1:FORY=0TO7:XS=5120+8*X+Y:POKEXS,PEEK(XS+8*CE):
    NEXT:FORY=0TO7                                  :rem 82
707 Z=Z+PEEK(5120+8*X+Y):NEXT                      :rem 218
710 IFZ=0ORZ=2040THENGOSUB760                       :rem 56
715 IF(X+CE)=(C-1)THENLU=X:X=C-1                   :rem 140
```

```
716 Z=Ø:NEXT                                                :rem 222
72Ø CH=C-1:C=LU+1                                           :rem 185
725 FORX=CTOCH:FORY=ØTO7:POKE512Ø+8*X+Y,Ø:NEXT:NEXT:D=PEEK(A)
    :POKEA,1:CL=256-C:RETURN                                :rem 218
76Ø POKE16347,-(Z=2Ø4Ø):POKE163Ø4,X:SYS163Ø1                :rem 247
77Ø CE=CE+1:X=X-1:RETURN                                    :rem 23Ø
8ØØ GOSUB48:PRINT"IS PRINTER READY?":PRINT"IF SO HIT ANY KEY"
                                                            :rem 245
8Ø1 GETA$:IFA$=""THEN8Ø1                                    :rem 85
8Ø3 GOSUB48:PRINT"PICK:(1) CURSOR CHAR":PRINTTAB(5);"(2) SIZE
     AS ABOVE":PRINTTAB(5);"{UP}START AT CURSOR"            :rem 8
8Ø4 OPEN4,4                                                 :rem 98
8Ø5 GETA$:ON-(A$<"1")-(A$>"2")GOTO8Ø5:ONVAL(A$)GOSUB81Ø,82Ø:C
    LOSE4:RETURN                                            :rem 7
81Ø PRINT#4,CHR$(14)"CURSOR CHARACTER:":PRINT#4,CHR$(15)
                                                            :rem 75
812 FORX=512Ø+D*8TO5127+D*8:PRINT#4,PEEK(X);:NEXT          :rem 191
813 PRINT#4:PRINT#4,"COLOR MEMORY=";PEEK(A+33792)AND15 :rem 1
814 PRINT#4,"AUXILIARY COLOR=";INT(PEEK(36878)/16):RETURN
                                                            :rem 18
82Ø POKEA,D:PRINT#4,CHR$(14)"CURSOR IN UPPER LEFT HAND CORNER
    ":PRINT#4,"OF A SQUARE";WI;                             :rem 26
821 PRINT#4,"X";TL;:PRINT#4,"(WIDTH X HEIGHT)":FORX=1TOTL:PRI
    NT#4,CHR$(15)"ROW #";X                                  :rem 235
824 FORY=1TOWI:PRINT#4,"CHARACTER #";Y;"-";                 :rem 72
825 FORZ=ØTO7:PRINT#4,PEEK(512Ø+Z+8*PEEK(A+Y-1+22*(X-1)));:NE
    XT:PRINT#4,"***":NEXT:NEXT                              :rem 2Ø4
83Ø PRINT#4:PRINT#4,"COLOR MEMORY":FORX=1TOTL:PRINT#4,"ROW #"
    ;X:FORY=1TOWI                                           :rem 246
832 PRINT#4,"CHARACTER #";Y;"-";                            :rem 28
835 PRINT#4,PEEK(A+33792+Y-1+(X-1)*22)AND15:NEXT:NEXT:rem 1Ø3
84Ø POKEA,1:PRINT#4,"AUXILIARY COLOR=";INT(PEEK(36878)/16):RE
    TURN                                                    :rem 24
9ØØ GOSUB48:PRINT"FILENAME? ";:B$="":FORX=ØTO254           :rem 144
9Ø5 GETA$:IFA$=""THEN9Ø5                                    :rem 95
91Ø IFA$=CHR$(13)THENX=254:GOTO92Ø                          :rem 238
915 B$=B$+A$:PRINTA$;                                       :rem 159
92Ø NEXT:POKEA,D:POKE5114,C:GOSUB48:PRINT"IS PROPER TAPE IN?"
    :PRINT"IF SO HIT ANY KEY"                               :rem 46
922 GETA$:IFA$=""THEN922                                    :rem 93
923 GOSUB48:OPEN1,1,1,B$:FORX=384ØØTO389Ø5:PRINT#1,PEEK(X)AND
    15:NEXT                                                 :rem 155
924 PRINT#1,INT((PEEK(36878))/16):CLOSE1                    :rem 54
925 POKE5115,PEEK(45):POKE5116,PEEK(46):POKE43,Ø:POKE44,18:PO
    KE45,Ø:POKE46,28:SAVECHR$(31),1,1                       :rem 234
935 POKE833,32:POKE43,1:POKE44,28:POKE45,PEEK(5115):POKE46,PE
    EK(5116)                                                :rem 21
94Ø POKEA,1:RETURN                                          :rem 132
1ØØØ GOSUB48:PRINT"FILENAME? ";:B$="":FORX=ØTO254          :rem 184
1Ø Ø5 GETA$:IFA$=""THEN1Ø Ø5                                :rem 175
1Ø1Ø IFA$=CHR$(13)THENX=254:GOTO1Ø2Ø                        :rem 62
```

```
1015 B$=B$+A$:PRINTA$;                                    :rem 199
1020 NEXT:GOSUB48:PRINT"IS PROPER TAPE IN?":PRINT"IF SO HIT A
     NY KEY"                                              :rem 153
1022 GETA$:IFA$=""THEN1022                                :rem 173
1023 GOSUB48:OPEN1,1,0,B$:FORX=38400TO38905:INPUT#1,CO:POKEX,
     CO:NEXT:INPUT#1,CO                                   :rem 171
1024 POKE36878,PEEK(36878)AND15OR(16*CO):CLOSE1:LOADCHR$(31):
     RETURN                                               :rem 17
1100 GOSUB48:PRINT"HIT P, E OR C TO SET{2 SPACES}PRINT, ERASE
      OR CHANGEMODES":FORX=1TO2000:NEXT                   :rem 127
1105 B$="C":SD%=0:SR%=0:POKEA,D:GOSUB83:IFA$="R"THEND=PEEK(A)
     :POKEA,1:RETURN                                      :rem 141
1110 CX=1:CY=1:ONSD+1GOSUB96                              :rem 201
1120 GETA$:IFA$=CHR$(13)THEND=PEEK(A):POKEA,1:RETURN :rem 178
1122 IF(A$="P")OR(A$="E")OR(A$="C")THENB$=A$              :rem 75
1125 POKE37154,127:ON(PEEK(37137)AND32)+1GOSUB1150        :rem 64
1130 A1=(PEEK(37137)AND28)OR(PEEK(37152)AND128):A1=ABS((A1-10
     0)/4)-7                                              :rem 173
1135 ONA1GOTO1136,1137,1138,,1139,1140,1144,,,,1141,1142,1143
                                                          :rem 60
1136 CX=CX+(CX>1):CY=CY-(CY<8*TL):GOTO1144                :rem 8
1137 CX=CX+(CX>1):CY=CY+(CY>1):GOTO1144                   :rem 56
1138 CX=CX+(CX>1):GOTO1144                                :rem 3
1139 CY=CY-(CY<8*TL):GOTO1144                             :rem 216
1140 CY=CY+(CY>1):GOTO1144                                :rem 255
1141 CX=CX-(CX<8*WI):GOTO1144                             :rem 206
1142 CX=CX-(CX<8*WI):CY=CY+(CY>1):GOTO1144                :rem 5
1143 CX=CX-(CX<8*WI):CY=CY-(CY<8*TL)                      :rem 154
1144 POKE37154,255:GOTO1120                               :rem 252
1150 X2=CX-1:X3=CY-1                                      :rem 131
1151 X4=A+INT(X2/8)+22*INT(X3/8):X5=PEEK(X4)*8+5120+X3-8*INT(
     X3/8)                                                :rem 223
1152 ON-(B$="P")-2*(B$="E")GOTO1154,1156                  :rem 30
1153 IF(PEEK(X5)AND(2↑(7-(X2-(INT(X2/8)*8)))))>0THEN1156
                                                          :rem 104
1154 POKEX5,PEEK(X5)OR2↑(7-(X2-(INT(X2/8)*8))):RETURN:rem 158
1156 POKEX5,PEEK(X5)AND(255-(2↑(7-(X2-(INT(X2/8)*8))))):RETUR
     N                                                    :rem 61
1200 MD$="ERASE":DM$="E":RETURN                           :rem 56
1300 MD$="PRINT":DM$="P":RETURN                           :rem 97
1400 GOSUB48                                              :rem 177
1410 PRINT"HIT COLOR KEYS TO SET COLOR. HIT {RVS}SHIFT{OFF} T
     HEN COLOR KEY TO SET MULTI-COLOR MODE.";             :rem 101
1415 FORX=1TO6000:NEXT                                    :rem 90
1420 GOSUB48:PRINT"ENTER A NUMBER FROM 0 TO 15 TO SET AUX COL
     OR";                                                 :rem 234
1430 GOSUB75:ON-(A$="DEL")GOTO1420:POKE36878,PEEK(36878)AND15
     OR(16*VAL(A$))                                       :rem 68
1499 GOSUB100:RETURN                                      :rem 2
30000 DATA162,0,169,170,221,0,18,240,36,144,42,224,255,208,26
      ,169,19,205,179,63,240,42                           :rem 179
```

```
30001 DATA162,0,169,19,141,179,63,141,222,63,141,228,63,141,2
      33,63,76,175,63,232,76,175                          :rem 248
30002 DATA63,169,170,157,0,18,76,184,63,188,0,18,136,152,157,
      0,18,76,184,63,169,18,141                           :rem 213
30003 DATA179,63,141,222,63,141,228,63,141,233,63,96   :rem 52
```

Griffin I. Johnson

Music Mate

Have you wanted to include music in your programs but found it too time-consuming and tedious to determine the frequency number and duration required for each note? "Music Mate" is a program that enables the user to program a tune with a minimum of time and effort through the use of visual prompts rather than with coded numbers. REMark statements have been omitted from the program to allow the maximum number of notes to be entered.

Once the first part of the program is loaded and run, the custom characters required for the main program are ready and Part 2 will be automatically loaded. Be sure to save both Program 1 and Program 2 before running either. Disk users should save Program 2 with the filename MATE. Tape users should delete lines 40000 to 40030 in Program 1 and add this line:

```
40000 POKE 198,5:POKE631,78:POKE632,69:POKE633,87:POKE634,13:POKE635,131
```

Be sure to save Program 2 on the same tape immediately following Program 1.

Writing Music

The program prompts the user to enter the total number of notes and rests for the tune up to a maximum of 60. Instructions for using the program are printed on the screen until any key is hit. Then, three columns of input choices are printed on the screen, including a message at the bottom of the screen indicating the note currently being entered.

The first column shows note and rest choices, for example, quarter note, half note, quarter rest, and so on. The second column is a small section of staff lines representing the treble clef from B-flat below middle C to two octaves above. The third column has only three choices—whether the note is natural, sharp, or flat.

The Music Mate program automatically positions the cursor to the left of the first column. Using the cursor up/down key, you can position the cursor to the left of the note or rest to be selected. You enter the note or rest duration by hitting the Z key. The one exception to this is a dotted note. For a dotted note you should position the cursor up one space and hit the Z key to enter the dot before entering the note. Also, at the bottom of the first column is the letter *R*. Choose *R* if you make a mistake in entering a note. To return to the note for correction, position the cursor to the left of the *R* and hit the Z key. This may be done repeatedly to return to any note.

The cursor them moves to the second column if a note was entered in the first column. Position the cursor to the line or space for each note by using the cursor up/down key. The program offers a two-octave range from B-flat below middle C to B above the staff. Hitting the Z key enters the note

and moves the cursor to the third column. In this column the cursor is positioned opposite the natural sign and must be moved up or down to select flat or sharp. Again, hitting the Z key enters the choice.

The above process is repeated for each note or rest. When all notes and rests have been entered, the program prompts with, "Do you want the music played? (Y/N)." If the choice is Y, the music is played as many times as desired. If the choice is N, the program prompts with, "Do you want the BASIC statements? (Y/N)." Entering Y prints the BASIC statements and DATA statements for the music which has been entered. These may then be copied to add music to your own programs.

Program 1. Music Mate

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 REM PART ONE OF TWO PART PROGRAM                          :rem 60
20 PRINT"{CLR}   MUSIC MATE"                                :rem 110
20000 POKE56,28:POKE52,28:POKE51,0:POKE55,0:CLR             :rem 253
20010 M1=7168:CS=32768                                      :rem 155
20020 FORI=M1TOM1+511:POKEI,PEEK(I+CS-M1):NEXT               :rem 68
20050 READX:IFX<0THEN20080                                    :rem 33
20060 FORI=XTOX+7:READJ:POKEI,J:NEXT                          :rem 91
20070 GOTO20050                                               :rem 41
20080 DATA7384,32,32,44,52,36,40,48,32                      :rem 129
20090 DATA7392,32,40,56,40,56,40,8,0                          :rem 25
20100 DATA7400,36,36,126,36,126,36,36,0                     :rem 167
20110 DATA7408,12,10,13,10,57,120,120,48                    :rem 200
20120 DATA7416,8,12,10,9,56,120,120,48                      :rem 115
20130 DATA7432,8,8,8,8,56,120,120,48                          :rem 29
20140 DATA7440,8,8,8,8,56,72,72,48                          :rem 201
20150 DATA7448,0,0,0,0,48,72,72,48                          :rem 179
20160 DATA7456,0,0,0,60,60,255,0,0                          :rem 161
20170 DATA7464,2,6,58,122,50,2,2,2                          :rem 174
20180 DATA7472,0,0,255,60,60,0,0,0                          :rem 161
20190 DATA7480,16,24,60,48,24,60,48,24                      :rem 134
20200 DATA7528,0,0,0,255,0,0,0,0                              :rem 48
20205 DATA7504,0,0,0,32,0,0,0,0                             :rem 248
20210 DATA7648,0,252,254,255,254,252,0,0                    :rem 220
20220 DATA-1                                                :rem 110
40000 POKE198,5:POKE631,78:POKE632,69:POKE633,87:POKE634,13:P
      OKE635,13                                               :rem 48
40005 POKE 36879,8:PRINT"{CLR}{YEL}PLEASE WAIT...{BLK}"
                                                            :rem 226
40010 PRINT"{5 DOWN}PO198,1:PO631,13:LOAD";CHR$(34);"MATE";CH
      R$(34);",8"                                           :rem 135
40020 PRINT "{4 DOWN}RUN"                                     :rem 0
40030 PRINT"{HOME}":END                                     :rem 236
```

Program 2. Mate

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 POKE36869,255:CLR                                          :rem 83
5 DIMN%(36),T%(60),NM%(60)                                   :rem 59
7 Z=0:X=0:S1=36876:TT%=400:M1=7168:CS=32768:L=1:SM=7680:V=368
  78                                                        :rem 125
10 FORI=1TO36:READN%(I):NEXT:POKE36879,110                   :rem 31
15 PRINT"{CLR}{WHT}ENTER NUMBER OF NOTES {DOWN}AND RESTS.
   {2 SPACES}(MAX.60){2 SPACES}{DOWN}":INPUTA               :rem 190
20 PRINT"{CLR}TO ENTER EACH NOTE OR {DOWN}REST MOVE CURSOR UP
    OR{DOWN}DOWN AND HIT Z."                                 :rem 2
25 PRINT:PRINT"RESTS REQUIRE ONLY ONE{DOWN}INPUT AND NOTES TH
   REE {DOWN}INPUTS. ENTER R TO{4 SPACES}"                  :rem 219
30 PRINT"REPEAT PREVIOUS NOTE.":PRINT                       :rem 157
31 PRINT:PRINT"{YEL}{5 RIGHT}HIT ANY KEY{WHT}"              :rem 230
32 GETS$:IFS$=""THEN32RUN                                    :rem 6
35 GOSUB1000                                                :rem 169
40 Z=Z+1:PRINT"{HOME}"TAB(255);TAB(158);"NOTE NUMBER "Z
                                                            :rem 108
50 B=90:G=1                                                   :rem 60
55 GOSUB2000                                                :rem 172
60 X=INT((B+R)/22)-3:IFX=0THENG=1.5                           :rem 15
65 IFX<1ORX>10THEN55                                        :rem 240
70 ONXGOTO100,110,120,130,140,150,160,170,180,185           :rem 252
100 T%(Z)=96/G:GOTO200                                      :rem 220
110 T%(Z)=48/G:GOTO200                                      :rem 218
120 T%(Z)=24/G:GOTO200                                      :rem 213
130 T%(Z)=12/G:GOTO200                                      :rem 211
140 T%(Z)=6/G:GOTO200                                       :rem 167
150 T%(Z)=48:GOTO190                                        :rem 112
160 T%(Z)=24:GOTO190                                        :rem 107
170 T%(Z)=12:GOTO190                                        :rem 105
180 T%(Z)=6:GOTO190                                           :rem 61
185 Z=Z-2:GOTO40                                            :rem 197
190 NM%(Z)=0:GOTO420                                        :rem 123
200 B=162:GOSUB2000                                           :rem 38
220 X=INT((B+R)/22):IFX<1ORX>16THEN200                      :rem 245
230 B=170:GOSUB2000                                           :rem 40
250 Y=INT((B+R)/22)-7:IFY<-1ORY>1THEN230                      :rem 89
260 ONXGOTO270,280,290,300,310,320,330,340,350,360 ,370,380,39
    0,400,410,415                                           :rem 214
270 NM%(Z)=N%(1+Y):GOTO420                                  :rem 195
280 NM%(Z)=N%(4+Y):GOTO420                                  :rem 199
290 NM%(Z)=N%(7+Y):GOTO420                                  :rem 203
300 NM%(Z)=N%(10+Y):GOTO420                                 :rem 237
310 NM%(Z)=N%(11+Y):GOTO420                                 :rem 239
320 NM%(Z)=N%(14+Y):GOTO420                                 :rem 243
330 NM%(Z)=N%(17+Y):GOTO420                                 :rem 247
340 NM%(Z)=N%(18+Y):GOTO420                                 :rem 249
350 NM%(Z)=N%(21+Y):GOTO420                                 :rem 244
```

```
360 NM%(Z)=N%(24+Y):GOTO420                                  :rem 248
370 NM%(Z)=N%(27+Y):GOTO420                                  :rem 252
380 NM%(Z)=N%(28+Y):GOTO420                                  :rem 254
390 NM%(Z)=N%(31+Y):GOTO420                                  :rem 249
400 NM%(Z)=N%(34+Y):GOTO420                                  :rem 244
410 NM%(Z)=N%(35+Y):GOTO420                                  :rem 246
420 IFZ=ATHEN450                                             :rem 197
430 GOTO40                                                    :rem 52
450 PRINT"{CLR}DO YOU WANT THE MUSIC PLAYED(Y/N)"             :rem 15
460 GETS$:IFS$=""THEN460                                     :rem 123
465 IFS$<>"N"ANDS$<> "Y"GOTO460                              :rem 229
470 IFS$="N"THEN590                                           :rem 61
475 POKEV,15                                                 :rem 183
480 FORI=1TOA:POKES1,NM%(I)                                  :rem 145
490 FORDT=1TO24*TT%/T%(I):NEXTDT:POKES1,0:NEXTI              :rem 232
495 POKEV,0:POKES1,0:GOTO450                                 :rem 216
590 PRINT"{CLR}DO YOU WANT THE BASIC STATEMENTS(Y/N)?"
                                                             :rem 125
600 GETS$:IFS$=""THEN600                                     :rem 115
610 IFS$<>"N"ANDS$<>"Y"THEN600                               :rem 207
615 C=11:B=INT(A/11+.95)                                      :rem 12
620 IFS$="N"THENEND                                          :rem 115
625 PRINT"2000POKE36878,15"                                     :rem 1
626 PRINT"2100S1=36876"                                         :rem 1
630 PRINT"2150T="TT%                                         :rem 144
640 PRINT"2200FORI=1TO"A":READD,N"                               :rem 9
660 PRINT"2300POKES1,N"                                       :rem 95
670 PRINT"2400FORDT=1TO24*T/D"                                :rem 27
675 PRINT"2500NEXTDT:POS1,0"                                  :rem 74
680 PRINT"2600NEXTI:PO36878,0"                               :rem 132
685 FORK=1TOB:L=1+(K-1)*11:READLN$                           :rem 225
690 IFK=BTHENC=A-(B-1)*11                                    :rem 146
695 PRINT:PRINTLN$;"DATA";:FORI=LTOL-1+C:PRINTT%(I)"{LEFT},";
    NM%(I)"{LEFT},";:NEXTI:PRINT"{LEFT} ":                   :rem 160
700 PRINT"{YEL}{5 RIGHT}HIT A KEY{WHT}"                      :rem 171
710 GETS$:IFS$=""THEN710                                     :rem 119
720 NEXTK                                                     :rem 35
750 POKE36869,240:END                                        :rem 174
900 DATA239,238,238,237,236,236,235,233,233,232,231,229,229,2
    28,227,227                                               :rem 244
910 DATA225,223,221,221,219,217,217,215,212,212,209,207,203,2
    03                                                        :rem 58
920 DATA201,199,199,195,191,187                               :rem 68
950 DATA2800,2900,3000,3100,3200,3300                         :rem 59
1000 PRINT"{CLR}":SC=38400                                    :rem 49
1100 PRINTTAB(47)"*{DOWN}{LEFT}↑{DOWN}←{DOWN}{2 LEFT}↓
     {2 DOWN}{LEFT}#{DOWN}%{DOWN}{2 LEFT}'{DOWN}&{DOWN}
     {2 LEFT}$${DOWN}R":POKE7838,34:POKE7838+SC,1          :rem 2
1150 PRINT"{HOME}"TAB(54)"-{2 DOWN}{2 LEFT}----{2 DOWN}
     {4 LEFT}----{2 DOWN}{4 LEFT}----{2 DOWN}{4 LEFT}----
     {2 DOWN}{4 LEFT}----{2 DOWN}{3 LEFT}-"                  :rem 11
```

```
1400 PRINT"{HOME}"TAB(150)"]{DOWN}£{DOWN}{2 LEFT}["   :rem 27
1600 RETURN                                          :rem 167
2000 R=0                                             :rem 129
2010 POKESM+B+R,60:POKESC+B+R,2                      :rem 85
2020 GETS$:IFS$=""THEN2020                           :rem 207
2025 IFS$="{DOWN}"THENPOKE38400+B+R,6:R=R+22         :rem 180
2030 IFS$="{UP}"THENPOKE38400+B+R,6:R=R-22           :rem 50
2040 IFS$="Z"THENPOKEB+R+SC,6:GOTO2070               :rem 35
2060 GOTO2010                                        :rem 196
2070 RETURN                                          :rem 169
```

Gene Klco

Piano Player

Many programs are available to emulate a piano on the VIC-20, but most do only that. "Piano Player" not only plays in any of the four voices of the VIC, and five combinations of them, but also displays a piano keyboard and the musical staff. Depressing a VIC key will display the note played on the staff and the piano keyboard as well as play the note. So extensive is the graphics in this program that a 3K expander is required and 64 programmable characters are used.

The objective of this program is to emulate a piano/organ/synthesizer, and to display the full graphics of the musical staff and the piano keyboard. Since the three voices the VIC plays overlap, I decided to use the different voices to provide the lower octave and not to display the bass clef. This left more characters available to optimize what was being used. For instance, 17 special characters were used to form the treble clef.

As the programs are listed below, Program 1 will load and run Program 2 if you use disk and save Program 2 with the filename PIANO.2. Tape users should delete lines 153, 155, and 156 of Program 1 and change line 154 to read

154 POKE198,1:POKE631,131:END

What to Do with It

Piano Player is in two parts to fit on the 3K expanded VIC. The first part of Piano Player loads the programmable characters and gives instructions after a short fanfare. Then, the second part prints the musical staff and the piano keyboard on the screen. Piano Player is set for voice 2 (treble) initially. To change voices, just depress one of the numbers from 1 to 9. Pressing 1 gives voice 1 (base), 2 gives voice 2 (treble), 3 gives voice 3 (soprano), and 4 gives voice 4 (noise). Keys 5–9 give combinations of the four voices. Try them for synthesized effects.

The middle row of the keyboard (A through =) plays the white keys from notes B to F (1-1/2 octaves). Middle C is the key S. The black piano keys are right above this row of VIC keys from Q to ↑.

Press a few keys. You will notice that the sound and graphics last as long as a key is held down. Since the VIC reads only one key at a time, you won't be able to play two keys at once.

Sample Songs

To familiarize yourself with Piano Player and its piano/synthesizer capabilities, try these simple songs (these are the VIC keys):

The slash (/) indicates a pause.

Happy Birthday

S S D S G F / S S D S H G / S S L J G F D / I I J G H G

My Country 'Tis of Thee

S S D A S D / F F G F D S / D S A S /
H H H H G F / G G G G F D /
F G F D S F G H / J G F D S

Saints Go Marching In

H K L : / H K L : / H K L : K H K J /
K K J H H K : : : L / K L : K J J H

Note Guide

Here's a conversion guide for VIC keys to notes:

VIC	Note	VIC	Note
Q	B♭	A	B
E	C♯	S	C
R	E♭	D	D
Y	F♯	F	E
U	G♯	G	F
I	B♭	H	G
P	C♯	J	A
@	E♭	K	B
↑	F♯	L	C
		:	D
		;	E
		=	F

Program 1. Piano Player

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}{4 DOWN}{RED}*********************"   :rem 193
11 PRINT"*********************"                    :rem 207
12 PRINT"** VIC PIANO PLAYER **"                      :rem 2
13 PRINT"*********************"                    :rem 209
14 PRINT"*********************"                    :rem 210
50 S1=36974:S2=36875:S3=36876:S4=36877:SB=36879:V=36878:E=25:
   Q=50:W=100:P=200                                   :rem 86
55 SX=SB:SY=S2:SZ=SB:F1=207:F2=195:GOSUB90           :rem 128
56 SX=S1:SY=S2:SZ=SB:F1=201:F2=192:GOSUB90           :rem 103
57 SX=SB:SY=S2:SZ=S3:F1=207:F2=195:GOSUB90           :rem 115
58 SX=S1:SY=SB:SZ=S3:F1=209:F2=207:GOSUB90           :rem 111
59 SX=SB:SY=S2:SZ=SB:F1=225:F2=219:GOSUB90           :rem 129
60 SX=S1:SY=S2:SZ=S3:F1=207:F2=219:POKESB,57:GOSUB90 :rem 239
61 FORD=15TO0STEP-1:POKEV,D:FORT=0TO25:NEXTT:NEXTD:POKESB,27:
   GOTO100                                           :rem 222
```

```
90 POKEV,0:POKESX,F1:POKESY,F1:POKESZ,F1:FORN=1TO3:POKEV,15
                                                    :rem 118
93 FORT=0TOQ:NEXTT:POKEV,0:FORR=0TOE:NEXTR:NEXTN     :rem 74
95 POKESX,F2:POKESY,F2:POKESZ,F2:POKESB,F2+5:POKEV,15:rem 178
97 FORT=0TOW:NEXTT:FORT=0TOP:NEXTT:RETURN           :rem 155
100 FORI=7168TO7223:POKEI,PEEK(I+25608):NEXT:FORI=7224TO7287:
    POKEI,PEEK(I+26312):NEXT                        :rem 187
102 FORI=7240TO7247:POKEI,255-PEEK(I+26304):NEXT     :rem 77
103 FORI=7272TO7279:POKEI,PEEK(I+26240):NEXT        :rem 142
104 FORI=7288TO7295:POKEI,255-PEEK(I+25736):NEXT    :rem 102
105 FORI=7296TO7303:POKEI,255-PEEK(I+25752)         :rem 225
106 FORJ=7304TO7679:READA:POKEJ,A:NEXT:POKE52,27:POKE56,27:PR
    INT"{CLR}"                                      :rem 239
110 PRINT"{BLK}{2 DOWN} WANT INSTRUCTIONS ?{DOWN}{6 LEFT}(Y/N
    )"                                               :rem 41
112 GETA$:IFA$=""THEN112                             :rem 75
114 IFA$="N"THEN153                                  :rem 33
116 PRINT"{CLR}{2 DOWN}USE YOUR VIC KEYBOARD {DOWN}AS A PIANO
     KEYBOARD:"                                     :rem 117
118 PRINT"{DOWN}THE VIC KEYS FROM 'A' {DOWN}TO '=' ARE THE WH
    ITE"                                            :rem 183
120 PRINT"{DOWN}KEYS FROM 'B' TO 'F', {DOWN}WITH 'S' AS MIDDL
    E C."                                           :rem 122
122 PRINT"{DOWN}THE BLACK KEYS ARE{4 SPACES}{DOWN}THE KEYS FR
    OM 'Q' TO"                                      :rem 109
124 PRINT"{DOWN}'↑'."                                :rem 83
126 PRINT"{DOWN}--- PRESS ANY KEY ---"              :rem 231
128 GETA$:IFA$=""THEN128                             :rem 89
130 PRINT"{CLR}{DOWN}THIS PROGRAM USES ALL {DOWN}THE VOICES O
    F THE VIC"                                      :rem 133
132 PRINT"{DOWN}-ALTO, SOPRANO, TENOR,{DOWN}AND NOISE."
                                                    :rem 147
134 PRINT"{DOWN}IT STARTS WITH SOPRANO{DOWN}THEN YOU CAN CHAN
    GE"                                              :rem 11
136 PRINT"{DOWN}TO ANY OTHER BY FIRST {DOWN}PRESSING '1' THRU
     '9'."                                          :rem 159
138 PRINT"{2 DOWN}FOR A CHART OF THE{4 SPACES}{DOWN}VOICES PR
    ESS ANY KEY"                                    :rem 215
140 GETA$:IFA$=""THEN140                             :rem 77
142 PRINT"{CLR}{DOWN}1=V1{7 SPACES}5=V1+V2{4 SPACES}{DOWN}2=V
    2{7 SPACES}6=V2+V3{4 SPACES}{DOWN}3=V3{7 SPACES}7=V1+V3"
                                                     :rem 29
144 PRINT"{DOWN}4=V4{7 SPACES}8=V1+V2+V3 {DOWN}9=V1+V2+V3+V4
    {9 SPACES}{2 DOWN}WHERE:"                       :rem 218
146 PRINT"{DOWN}V1=ALTO{5 SPACES}V3=TENOR{2 SPACES}{DOWN}V2=S
    OPRANO{2 SPACES}V4=NOISE{2 SPACES}{DOWN}--- PRESS ANY KEY
     ---"                                           :rem 121
148 GETA$:IFA$=""THEN148                             :rem 93
153 POKE 36879,15:PRINT"{CLR}{YEL}PLEASE WAIT....."  :rem 124
154 POKE198,2:POKE 631,13:POKE 632,13                :rem 240
```

```
155 PRINT "{HOME}{BLK}":PRINT "{2 DOWN}LOAD";CHR$(34);"PIANO.
    2";CHR$(34);",8":                                  :rem 183
156 PRINT "{4 DOWN}RUN":PRINT"{HOME}":END              :rem 213
1000 DATA15,14,12,14,12,14,15,15                        :rem 54
1010 DATA240,176,16,176,16,176,240,240                 :rem 122
1020 DATA239,239,239,227,237,237,235,231               :rem 238
1030 DATA14,14,14,14,14,14,14,14:REM#20                :rem 221
1040 DATA240,240,240,48,208,208,176,112                :rem 167
1050 DATA0,0,1,1,1,1,1,255                               :rem 5
1060 DATA0,1,1,3,7,14,28,255                           :rem 124
1070 DATA56,48,113,97,99,99,99,255                     :rem 228
1080 DATA97,48,48,24,12,6,3,255:REM#25                 :rem 213
1090 DATA0,0,0,24,60,118,99,227                         :rem 27
1100 DATA195,193,129,129,131,131,135,255               :rem 227
1110 DATA135,135,79,94,94,60,124,255                    :rem 38
1120 DATA240,240,208,144,16,8,8,255                    :rem 228
1130 DATA127,255,200,132,132,4,4,255:REM#30            :rem 175
1140 DATA130,194,34,2,1,1,255,255                      :rem 121
1150 DATA0,0,0,0,56,124,252,216                          :rem 9
1160 DATA128,192,99,28,0,0,0,0                         :rem 233
1170 DATA0,128,128,128,128,128,128,255                 :rem 133
1180 DATA224,240,120,56,28,12,12,255:REM#35            :rem 185
1190 DATA12,12,24,24,48,32,192,255                     :rem 180
1200 DATA128,128,128,128,64,64,64,64                    :rem 37
1210 DATA64,128,0,0,0,0,0,0                             :rem 55
1220 DATA1,63,127,255,254,124,0,0                      :rem 117
1230 DATA1,63,127,255,254,124,0,255:REM#40             :rem 135
1240 DATA254,124,0,0,0,0,0,0                           :rem 103
1250 DATA254,124,0,0,0,0,0,255                         :rem 212
1260 DATA1,1,1,1,1,63,127,255                          :rem 171
1270 DATA1,1,1,1,1,1,1,255                              :rem 11
1280 DATA0,0,0,0,1,1,1,255:REM#45                      :rem 178
1290 DATA0,62,127,255,254,252,128,255                   :rem 83
1300 DATA0,0,0,0,0,62,127,255                          :rem 160
1310 DATA254,254,128,128,128,128,128,255               :rem 236
1320 DATA128,128,128,128,128,128,128,255               :rem 237
1330 DATA128,128,128,0,0,0,0,0:REM #50                 :rem 124
1340 DATA128,128,128,0,0,0,0,255                        :rem 67
1350 DATA20,62,20,20,20,62,20,255                      :rem 109
1360 DATA0,0,0,0,20,62,20,20                           :rem 102
1370 DATA20,62,20,0,0,0,0,0                             :rem 53
1380 DATA0,0,0,0,20,62,20,255:REM#55                    :rem 77
1390 DATA20,62,20,0,0,0,0,255                          :rem 163
1400 DATA0,0,0,32,32,32,32,32                          :rem 156
1410 DATA60,50,34,34,36,40,48,32                        :rem 75
1420 DATA32,32,32,32,60,50,34,255                      :rem 119
1430 DATA36,40,48,32,0,0,0,0:REM#60                     :rem 27
1440 DATA36,40,48,32,0,0,0,255                         :rem 225
1450 DATA0,0,0,32,32,32,32,255                         :rem 216
1460 DATA60,50,34,34,36,40,48,255                      :rem 135
```

Program 2. Piano.2

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
200 POKE 36879,27:PRINT"{CLR}"                                  :rem 5
210 SM=7680:CM=38400                                           :rem 75
230 S1=36874:S2=36875:S3=36876:S4=36877:V=36878:SB=36879:SX=2
                                                              :rem 209
240 DEFFNA(X)=PEEK(197)=X:YS=SM+318:YN=SM+406:POKEV-9,255
                                                              :rem 198
250 DIMA(47):FORN=0TO46:READA(N):NEXTN                         :rem 84
310 FORW=SMTOSM+505:POKEW,7:NEXT:FORQ=CMTOCM+505:POKEQ,0:NEXT
                                                               :rem 25
320 FOR X=SM+44TOSM+153:POKEX,11:NEXT                          :rem 19
322 Y=23:POKESM+44,22:FORX=SM+88TOSM+132STEP22:POKEX,Y:Y=Y+1:
    NEXTX                                                     :rem 239
324 Y=26:FORX=SM+23TOSM+177STEP22:POKEX,Y:Y=Y+1:NEXTX:rem 198
326 Y=35:POKESM+46,34:FORX=SM+112TOSM+178STEP22:POKEX,Y:Y=Y+1
    :NEXTX                                                     :rem 41
328 T=SM+264:M=SM+351:B=SM+439:FORK=T-22TOT-1:POKEK,11:NEXT
                                                              :rem 127
330 PRINT"{BLK}{HOME}{12 DOWN}"                               :rem 214
332 C=C+1:PRINT"{UP}OIHOGLOIHMIHOIHMIHOGLO"                   :rem 194
334 IFC<4GOTO332                                              :rem 173
336 C=C+1:PRINT"{UP}MGLMGLMGLMGLMGLMGLMGLM"                   :rem 196
338 IFC<8GOTO336                                              :rem 185
340 FORK=B+1TOB+22:POKEK,10:NEXT                              :rem 210
344 A=22:B=44:C=66:D=88:E=23                                   :rem 16
350 GETA$:IFA$=""THEN350                                       :rem 83
355 N=ASC(A$):IFN<48ORN>94THEN350                             :rem 213
360 F=A(N-48):IFF>0ANDF<10THENSX=F:GOTO350                     :rem 48
370 IFF=0THEN350                                              :rem 163
371 IFSX=1THENPOKES1,F:POKES2,0:POKES3,0:POKES4,0:POKEV,15:GO
    TO400                                                     :rem 208
372 IFSX=2THENPOKES1,0:POKES2,F:POKES3,0:POKES4,0:POKEV,15:GO
    TO400                                                     :rem 210
373 IFSX=3THENPOKES1,0:POKES2,0:POKES3,F:POKES4,0:POKEV,15:GO
    TO400                                                     :rem 212
374 IFSX=4THENPOKES1,0:POKES2,0:POKES3,0:POKES4,F:POKEV,15:GO
    TO400                                                     :rem 214
375 IFSX=5THENPOKES1,F:POKES2,F:POKES3,0:POKES4,0:POKEV,15:GO
    TO400                                                     :rem 238
376 IFSX=6THENPOKES1,0:POKES2,F:POKES3,F:POKES4,0:POKEV,15:GO
    TO400                                                     :rem 240
377 IFSX=7THENPOKES1,F:POKES2,0:POKES3,F:POKES4,0:POKEV,15:GO
    TO400                                                     :rem 242
378 IFSX=8THENPOKES1,F:POKES2,F:POKES3,F:POKES4,0:POKEV,15:GO
    TO400                                                      :rem 10
379 IFSX=9THENPOKES1,F:POKES2,F:POKES3,F:POKES4,F:POKEV,15:GO
    TO400                                                      :rem 34
395 GOTO350                                                   :rem 114
400 IFA$="A"THEN600                                            :rem 15
```

```
405 IFA$="Q"THEN610                                          :rem 37
410 IFA$="S"THEN620                                          :rem 36
415 IFA$="E"THEN630                                          :rem 28
420 IFA$="D"THEN640                                          :rem 24
425 IFA$="R"THEN650                                          :rem 44
430 IFA$="F"THEN660                                          :rem 29
435 IFA$="G"THEN670                                          :rem 36
440 IFA$="Y"THEN680                                          :rem 51
445 IFA$="H"THEN690                                          :rem 40
450 IFA$="U"THEN700                                          :rem 41
455 IFA$="J"THEN710                                          :rem 36
460 IFA$="I"THEN720                                          :rem 32
465 IFA$="K"THEN730                                          :rem 40
470 IFA$="L"THEN740                                          :rem 38
475 IFA$="P"THEN750                                          :rem 48
480 IFA$=":"THEN760                                          :rem 23
485 IFA$="@"THEN770                                          :rem 35
490 IFA$=";"THEN780                                          :rem 27
495 IFA$="="THEN790                                          :rem 35
500 IFA$="↑"THEN800                                          :rem 47
510 GOTO350                                                 :rem 103
600 GOSUB605                                                :rem 177
602 IFFNA(17)THEN602                                        :rem 124
604 GOSUB608:POKEY-A,7:POKEY,7:GOTO350                      :rem 123
605 Y=SM+180:K=6:N=1:GOSUB606:RETURN                        :rem 243
606 POKEYN-K,N:POKEY-C,B:POKEY-B,B:POKEY-A,B:POKEY,39:RETURN
                                                            :rem 181
608 POKEV,0:POKEYN-K,7:FORX=3TO1STEP-1:POKEY-A*X,11:NEXT:RETU
    RN                                                      :rem 148
610 GOSUB605:POKEYS-K-1,19:POKEY-E,57:POKEY-1,58            :rem 184
612 IFFNA(48)THEN612                                        :rem 130
614 GOSUB608:POKEY-A,7:POKEY,7:POKEY-E,7:POKEY-1,7:POKEYS-K-1
    ,15:GOTO350                                              :rem 19
620 GOSUB625                                                :rem 181
622 IFFNA(41)THEN622                                        :rem 125
624 GOSUB628:POKEY-E,7:POKEY-A,7:POKEY-A+1,7:POKEY,7:GOTO350
                                                              :rem 5
625 Y=SM+181:K=5:N=2:GOSUB626:POKEY-E,11:POKEY-A+1,11:RETURN
                                                            :rem 212
626 POKEYN-K,N:POKEY-D,45:POKEY-C,B:POKEY-B,B:POKEY-A,43:POKE
    Y,41:RETURN                                             :rem 157
628 POKEV,0:POKEYN-K,12:FORX=4TO1STEP-1:POKEY-A*X,11:NEXT:RET
    URN                                                     :rem 195
630 GOSUB625:POKEYS-K+1,16:POKEY-E,55:POKEY-1,54            :rem 177
632 IFFNA(49)THEN632                                        :rem 135
634 GOSUB628:POKEY-A,7:POKEY,7:POKEY-A+1,7:POKEYS-K+1,15:POKE
    Y-E,7:POKEY-1,7:GOTO350                                   :rem 4
640 Y=SM+161:K=3:N=3:GOSUB606                               :rem 214
642 IFFNA(18)THEN642                                        :rem 133
644 GOSUB608:POKEY,7:GOTO350                                :rem 236
650 GOSUB665:GOSUB656                                        :rem 23
```

```
652 IFFNA(10)THEN652                                        :rem 127
654 GOSUB628:POKEY,7:GOSUB658:GOTO350                        :rem 76
656 POKEY-E,59:POKEY-1,60:POKEYS-K-1,20:POKEYS-K,21:RETURN
                                                            :rem 150
658 POKEY-E,11:POKEY-1,7:POKEYS-K-1,9:POKEYS-K,8:RETURN:rem 9
660 GOSUB665                                                :rem 189
662 IFFNA(42)THEN662                                        :rem 134
664 GOSUB628:POKEY,7:GOTO350                                :rem 240
665 Y=SM+162:K=2:N=4:GOSUB626:RETURN                        :rem 250
670 GOSUB675                                                :rem 191
672 IFFNA(19)THEN672                                        :rem 140
674 GOSUB608:POKEY,11:GOTO350                                :rem 26
675 Y=SM+142:K=0:N=5:GOSUB606:POKEY,40:RETURN                :rem 72
680 GOSUB675:POKEY-1,52:POKEYS,17:POKEYS+1,18                :rem 34
682 IFFNA(11)THEN682                                        :rem 134
684 GOSUB608:POKEY,11:POKEY-1,11:POKEYS,9:POKEYS+1,8:GOTO350
                                                             :rem 24
690 GOSUB695                                                :rem 195
692 IFFNA(43)THEN692                                        :rem 141
694 GOSUB628:POKEY,11:GOTO350                                :rem 30
695 Y=SM+143:K=-1:N=6:GOSUB626:POKEY,42:RETURN              :rem 126
700 GOSUB695:POKEYS+2,16:POKEY-E,55:POKEY-1,56               :rem 65
702 IFFNA(51)THEN702                                        :rem 124
704 GOSUB628:POKEY,11:POKEYS+2,15:POKEY-E,11:POKEY-1,11:GOTO3
    50                                                      :rem 138
710 Y=SM+123:K=-3:N=0:GOSUB606:POKEY,40                      :rem 78
712 IFFNA(20)THEN712                                        :rem 122
714 GOSUB608:POKEY,11:GOTO350                                :rem 21
720 GOSUB735:POKEYS+K-1,20:POKEYS+K,21:POKEY-1,59:POKEY+A-1,6
    1                                                        :rem 34
722 IFFNA(12)THEN722                                        :rem 125
724 J=2:GOSUB738:GOSUB728:POKEYS+K-1,9:POKEYS+K,8:POKEY-1,11:
    POKEY+A-1,11:GOTO350                                     :rem 29
726 POKEY+C,12:POKEY+D,50:RETURN                              :rem 0
728 POKEY+C,7:POKEY+D,7:RETURN                              :rem 168
730 GOSUB735                                                :rem 185
732 IFFNA(44)THEN732                                        :rem 132
734 J=2:GOSUB738:GOSUB728:GOTO350                            :rem 25
735 Y=SM+102:K=4:N=1:GOSUB736:GOSUB726:RETURN                :rem 76
736 POKEYN+K,N:POKEY,47:POKEY+A,48:POKEY+B,49:RETURN :rem 103
738 POKEV,0:POKEYN+K,12:FORX=0TOJ:POKEY+A*X,11:NEXT:RETURN
                                                             :rem 60
740 GOSUB745                                                :rem 187
742 IFFNA(21)THEN742                                        :rem 129
744 GOSUB748:POKEY+C,7:GOTO350                               :rem 96
745 Y=SM+104:K=6:N=2:GOSUB746:POKEY+C,12:RETURN             :rem 185
746 POKEYN+K,N:POKEY,46:POKEY+A,49:POKEY+B,49:RETURN :rem 104
748 POKEV,0:POKEYN+K,7:FORX=0TO2:POKEY+A*X,11:NEXT:RETURN
                                                            :rem 249
750 GOSUB745:POKEYS+K,17:POKEYS+K+1,18:POKEY-1,52            :rem 10
```

```
752 IFFNA(13)THEN752                                          :rem 132
754 GOSUB748:POKEY+C,7:POKEYS+K,9:POKEYS+K+1,8:POKEY-1,11:GOT
    O350                                                       :rem 74
760 Y=SM+83:K=7:N=3:GOSUB736:POKEY+C,49:POKEY+D,50             :rem 63
762 IFFNA(45)THEN762                                          :rem 139
764 J=3:GOSUB738:POKEY+D,7:GOTO350                             :rem 86
770 GOSUB785:K=9:POKEYS+K-1,19:POKEY-E,62:POKEY-1,63 :rem 185
772 IFFNA(53)THEN772                                          :rem 140
774 GOSUB748:POKEY+C,11:POKEYS+K-1,15:POKEY-E,11:POKEY-1,11:G
    OTO350                                                    :rem 121
780 GOSUB785                                                  :rem 195
782 IFFNA(22)THEN782                                          :rem 138
784 GOSUB748:POKEY+C,11:GOTO350                               :rem 143
785 Y=SM+85:K=9:N=4:GOSUB746:POKEY+C,49:RETURN                :rem 164
790 GOSUB795                                                  :rem 197
792 IFFNA(46)THEN792                                          :rem 146
794 J=4:GOSUB738:GOTO350                                      :rem 198
795 Y=SM+64:K=10:N=5:GOSUB736:POKEY+C,49:POKEY+D,51:RETURN
                                                              :rem 141
800 GOSUB795:POKEYS+K+1,16:POKEY-1,55:POKEY+A-1,56             :rem 16
802 IFFNA(54)THEN802                                          :rem 129
804 J=4:GOSUB738:POKEYS+K+1,15:POKEY-1,11:POKEY+A-1,11:GOTO35
    0                                                         :rem 255
2000 DATA0,1,2,3,4,5,6,7,8,9                                  :rem 117
2010 DATA228,231,0,232,0,0,229,191,0,0                         :rem 90
2020 DATA201,199,207,209,215,221,219,223,225,0                :rem 253
2030 DATA0,0,227,187,203,195,0,217,0,0                        :rem 100
2040 DATA0,212,0,0,0,0,233                                      :rem 5
3000 END                                                      :rem 154
```

Paul
McManamon

Bitmapping Pixel

This drawing program for a VIC with 8K or more expansion and a joystick uses the bitmap screen to test your artistic talent. Just type in Program 1 and save it to disk or tape. Then type in Program 2 and save it. Disk users must save Program 2 with the filename BM PIXEL.2. Tape users should save Program 2 on the same tape immediately following Program 1. Tape users must also delete lines 15 and 20 from Program 1 and substitute the following for line 10:

10 POKE8192,0:POKE44,32:POKE643,32:POKE198,1:POKE631,131

Drawing Pixel by Pixel

Once you have the programs saved, run Program 1. It will change the location of BASIC and automatically load and run Program 2. The first thing you'll see is the character set printed to the screen in what appears to be random order. Then the whole screen is erased and changed to the bitmapped screen which is also erased. Finally, a single pixel dot will appear in the middle of the screen. Use a joystick to draw, one pixel at a time. It you wish to move the drawing position to another part of the screen without leaving a line, simply hold down the button while pushing on the stick.

Program 1. Bitmap Pixel

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 REM THIS PROGRAM MOVES BASIC                                  :rem 85
10 POKE8192,0:POKE44,32:POKE643,32:POKE198,2:POKE631,13:POKE
   {SPACE}632,13                                                :rem 73
15 PRINT "{CLR}":PRINT "{2 DOWN}LOAD";CHR$(34);"BM PIXEL.2";C
   HR$(34);",8":PRINT "{5 DOWN}RUN"                             :rem 39
20 PRINT "{HOME}"                                               :rem 70
```

Program 2. BM Pixel.2

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 REM MOVES SCREEN MEMORY                                       :rem 60
10 POKE36866,150:POKE36869,240                                  :rem 154
20 POKE 209,0:POKE210,30:POKE201,0:POKE202,0:POKE211,0:POKE21
   4,0                                                          :rem 121
30 POKE648,30:POKE244,150                                       :rem 136
90 DD=37154:P1=37151:P2=37152                                   :rem 38
99 REM SET UP SCREEN                                            :rem 167
100 POKE36879,107:PRINTCHR$(147)                                :rem 74
102 POKE36867,PEEK(36867)OR1                                    :rem 82
104 FORI=0TO21:POKE7680+I,220:POKE7922+I,220:POKE7944+I,220:N
    EXT                                                         :rem 9
```

```
106 FORI=0TO21:POKE38400+I,3:POKE38642+I,3:POKE38664+I,3:NEXT
                                                    :rem 120
108 FORI=0TO15:POKE7616+I,255:NEXT                   :rem 49
110 FOR L=0TO21:FORM=0TO9                           :rem 239
120 POKE7702+M*22+L,L*10+M                          :rem 138
130 NEXT:NEXT                                        :rem 76
140 POKE36869,252                                   :rem 153
150 FORI=4096TO7615:POKEI,0:NEXTI                    :rem 28
160 REM SCREEN IS{5 SPACES}Y=160 HIGH BY X=176{4 SPACES}WIDE
                                                     :rem 27
170 X=88:Y=80                                       :rem 213
200 GOSUB1000                                       :rem 211
210 IFJ1=1THENY=Y+1                                 :rem 133
220 IF J3=1THEN Y=Y-1                               :rem 138
230 IFY>159 THEN Y=159                              :rem 190
240 IFY<0THEN Y=0                                   :rem 223
250 IFJ0=1THENX=X+1                                 :rem 134
260 IFJ2=1THENX=X-1                                 :rem 139
270 IFX>175THENX=175                                :rem 188
280 IFX<0THENX=0                                    :rem 225
290 CH=INT(X/8)*10+INT(Y/16)                         :rem 63
300 RO=(Y/16-INT(Y/16))*16                          :rem 154
310 BY=4096+16*CH+RO                                 :rem 82
320 BI=7-(X-(INT(X/8)*8))                            :rem 69
330 IFFB=0THENPOKEBY,PEEK(BY)OR(2↑BI)                :rem 93
335 IFFB>.5THENPOKEBY,PEEK(BY)AND(255-(2↑BI))       :rem 226
340 GO TO 200                                        :rem 98
1000 REM SUBROUTINE TO READ JOY SWITCHES            :rem 208
1010 POKEDD,127:P=PEEK(P2)AND128                    :rem 108
1020 J0=-(P=0)                                      :rem 181
1030 POKEDD,255:P=PEEK(P1)                            :rem 1
1040 J1=-((PAND8)=0)                                 :rem 20
1050 J2=-((PAND16)=0)                                :rem 69
1060 J3=-((PAND4)=0)                                 :rem 20
1070 FB=-((PAND32)=0)                                :rem 81
1080 RETURN                                         :rem 169
```

Soori
Sivakumaran

Graph

"Graph" will allow you to create bar graphs on your VIC. A maximum of 16 items can be graphed, and the bars can be of eight different colors. Once a graph is created, it can be stored on cassette tape and later reproduced on the VIC's screen.

When you create a graph, you must give it a title. The graph will be referred to by this title when it's recalled if you save it on tape.

Using the graph program is easy. Suppose you want to graph the monthly fuel costs for a year. Enter the number 12 so the VIC will know how many items to graph. The total number of items that can be graphed is limited to 16.

The bars drawn by the program are a minimum width of one column. However, if fewer than 16 items are graphed, the program adjusts the width of the bars to a maximum number of columns while maintaining their uniformity.

You can set the vertical axis maximum value or the value of the vertical axis increments. Since the number of vertical divisions is set at 20 in the program, entering a vertical maximum of 100 would be identical to setting the vertical increments to 5.

The top row of the screen is reserved for the title of the graph. The next 20 rows of the screen are used for drawing the bars of the graph. The row below the bars is occupied by the horizontal axis. The last row of the VIC's screen contains the letters that label the horizontal axis. Thus, the entire 23 rows of the screen are occupied by the graph.

The program requires the user to identify the vertical axis, that is, what the numbers along the vertical axis represent. In the case of a graph of the year's fuel costs, the vertical axis would be identified as "Cost in Dollars."

Lines 177–198 of the program contain a FOR-NEXT loop in which the name, bar color, and value for each item are entered starting with item A. In the example of graphing a year's fuel costs, the name of item A would be 01 for January. Select the color of each bar by entering the number that corresponds with the color wanted according to the color chart that is printed on the screen. If the bar representing January's fuel costs was to be made yellow, the number 7 would be entered. (Remember that a white bar won't be distinguishable on a white screen.) The value of the item is the number of units of the item. For the graph of a year's fuel costs, the value 85 would be entered for item A if January's fuel cost was $85.

Lines 210–280 contain two FOR-NEXT loops. The first prints the vertical axis and the second labels it. Line 265 of the program limits the length of the numbers labeling the vertical axis to prevent them from running into the bars of the graph. If a number that will go along the vertical axis is longer than

two digits, the program may produce an OUT OF MEMORY error. Line 290 of the program draws the horizontal axis of the graph.

After the graph is completely printed on the screen, the routine in lines 620–720 scans the keyboard. If a letter that is a horizontal label is pressed, the name and value of the corresponding item is printed at the top left of the screen. If the V key is pressed, the vertical axis is identified.

Pressing the S key will save the graph on tape or disk. Disk users should change lines 60 and 2010 to:

60 OPEN 1,8,2,A$
2010 OPEN 1,8,2,A$+",S,W"

To reproduce a previously saved graph on the screen, simply run the program and select to display a stored graph.

"Graph" is quite a useful program for graphing all types of information, particularly in the area of budgeting.

Graph

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 B$="ABCDEFGHIJKLMNOP":H$=CHR$(112):I$=CHR$(145):J$=CHR$(17)
  :K$=CHR$(147):L$=CHR$(19)                                  :rem 83
7 M$=CHR$(18):N$=CHR$(146):O$=CHR$(157)                      :rem 79
10 PRINTK$;J$;J$;J$;J$TAB(3)M$"DO YOU WISH TO"N$            :rem 241
15 PRINTJ$TAB(2)"A)CREATE A NEW GRAPH"                      :rem 186
20 PRINTTAB(2)"B)DISPLAY A STORED":PRINT"GRAPH"             :rem 157
25 PRINTJ$"SELECT "M$;"A"N$" OR "M$"B"N$;                   :rem 146
30 INPUTF$                                                   :rem 93
35 IFF$="A"THEN100                                          :rem 227
40 PRINT"ENTER THE GRAPH'S":PRINT"TITLE";                    :rem 72
45 INPUTA$                                                   :rem 94
50 PRINT"OPENING FILE"                                      :rem 102
55 REM LOAD GRAPH                                           :rem 224
60 OPEN 1,1,0,A$                                             :rem 19
65 INPUT#1,A$                                               :rem 224
67 INPUT#1,G$                                               :rem 232
70 INPUT#1,A                                                :rem 184
75 INPUT#1,B                                                :rem 190
80 DIMD$(A):DIMC(A):DIMD(A)                                  :rem 15
85 FORI=1TOA                                                :rem 239
87 INPUT#1,D$(I)                                            :rem 129
90 INPUT#1,C(I)                                              :rem 86
92 INPUT#1,D(I)                                              :rem 89
95 NEXT I                                                   :rem 246
97 CLOSE 1:GOTO200                                           :rem 28
100 PRINTK$                                                 :rem 141
110 PRINTM$"GRAPH'S TITLE"N$;                               :rem 239
120 INPUTA$                                                 :rem 136
125 PRINTJ$"MAXIMUM OF 16 ITEMS"                            :rem 115
130 PRINTJ$"HOW MANY ITEMS TO"                               :rem 27
```

```
140 PRINT"GRAPH";                                          :rem 19
150 INPUT A                                               :rem 103
151 PRINTJ$"DO YOU WISH TO:"J$                            :rem 236
152 PRINT"1)SET VERTICAL AXIS":PRINT"MAXIMUM"             :rem 103
154 PRINT"2)SET VERTICAL":PRINT"INCREMENTS"                :rem 15
155 PRINTJ$;M$"SELECT (1 OR 2)"N$;:INPUTN                  :rem 96
157 IFN=1THEN163                                          :rem 177
158 PRINTJ$"THERE ARE 20 VERTICAL":PRINT"DIVISIONS."      :rem 218
160 PRINTJ$"INPUT VERTICAL":PRINT"INCREMENTS";            :rem 254
162 INPUTB:GOTO165                                        :rem 122
163 PRINTJ$"INPUT VERTICAL":PRINT"MAXIMUM";                :rem 39
164 INPUTN:B=N/20                                          :rem 17
165 PRINTK$;J$"IDENTIFY VERTICAL":PRINT"AXIS";:INPUTG$
                                                          :rem 235
166 PRINTJ$SPC(6)M$"BAR COLOURS"N$;J$                     :rem 209
167 PRINTSPC(2)"BLACK=0"SPC(4)" WHITE=1"                   :rem 64
168 PRINTSPC(4)"RED=2"SPC(6)"CYAN=3"                      :rem 113
170 PRINTSPC(1)"PURPLE=4"SPC(5)"GREEN=5"                  :rem 173
172 PRINTSPC(3)"BLUE=6"SPC(4)"YELLOW=7"                   :rem 111
173 DIMD$(A):DIMC(A):DIMD(A)                               :rem 66
175 PRINTJ$;M$"IDENTIFY ITEMS"N$;J$                       :rem 129
177 FORI=1TOA                                              :rem 33
180 PRINT"NAME ITEM "M$;MID$(B$,I,1)N$;                   :rem 154
182 INPUTD$(I)                                             :rem 45
185 PRINT"ITEM "M$;MID$(B$,I,1);N$" BAR COLOUR";          :rem 166
187 INPUT C(I)                                             :rem 13
188 PRINT"VALUE OF ITEM "M$;MID$(B$,I,1)N$;               :rem 147
190 INPUTD(I)                                               :rem 8
192 IFD(I)>(20*B)THEN190                                   :rem 47
195 PRINTL$:FORO=1TO6:PRINTJ$:NEXTO:FORJ=7966TO8098:POKEJ,32:
    NEXTJ                                                  :rem 87
198 NEXTI                                                  :rem 42
200 PRINTK$;J$;J$;J$TAB(3)"TO GET AN ITEM'S ":PRINT" NAME AND
     VALUE ENTER"                                         :rem 221
202 PRINTTAB(3)"THE LETTER THAT":PRINT" REPRESENTS THE ITEM."
                                                          :rem 246
204 PRINTJ$" ENTERING "M$"S"N$"WILL SAVE"SPC(3)"THE GRAPH ON
    {SPACE}TAPE."                                          :rem 76
205 PRINTJ$TAB(4)"ENTERING "M$"X"N$"WILL ":PRINTTAB(5)"END PR
    OGRAM."                                                :rem 14
206 PRINTJ$" ENTER "M$"V"N$"TO IDENTIFY"SPC(6)"VERTICAL AXIS"
                                                           :rem 85
207 PRINTJ$"ENTER ANY LETTER TO":PRINT"CONTINUE";          :rem 92
208 INPUTF$:PRINTK$                                       :rem 202
209 REM DRAW AXIS                                         :rem 226
210 FORI=1TO20                                             :rem 54
220 PRINTTAB(4)H$                                         :rem 233
230 NEXTI                                                  :rem 29
240 PRINTTAB(2)"0"                                        :rem 241
250 PRINTL$                                               :rem 148
260 FORI=0TO19                                             :rem 66
```

```
265 C$=STR$((B*20)-(B*I)):C$=LEFT$(C$,5)          :rem 71
270 PRINTO$;C$                                    :rem 59
280 NEXTI                                         :rem 34
290 FORI=1TO17:PRINTTAB(3+I)H$;I$:NEXTI           :rem 176
300 G%=16/A:DIME(16):DIMF(16):H=1:DIML(16)        :rem 62
310 FORI=1TOA                                     :rem 22
320 FORJ=1TOG%                                    :rem 67
330 E(H)=INT(D(I)/B)                              :rem 60
340 K=(D(I)/B)-E(H)                               :rem 202
350 F(H)=98                                       :rem 37
360 IFK>.75THENF(H)=160                           :rem 45
370 IFK<.25THENF(H)=32                            :rem 245
380 L(H)=C(I)                                     :rem 154
390 H=H+1                                         :rem 197
400 NEXTJ                                         :rem 29
410 NEXTI                                         :rem 29
415 H=H-1                                         :rem 197
417 REM DRAW BARS                                 :rem 214
420 FORI=HTO1STEP-1                               :rem 185
430 FORJ=1TOE(I)                                  :rem 184
435 IFE(I)=0THEN470                               :rem 65
440 POKE8146+I-(J*22),160                         :rem 39
450 POKE38866+I-(J*22),L(I)                       :rem 179
460 NEXTJ                                         :rem 35
470 POKE8146+I-(J*22),F(I)                        :rem 115
480 POKE38866+I-(J*22),L(I)                       :rem 182
500 NEXTI                                         :rem 29
505 H=1:PRINT                                     :rem 23
510 FORI=1TOA                                     :rem 24
520 FORJ=1TOG%                                    :rem 69
530 PRINTSPC(4+H)MID$(B$,I,1)I$                   :rem 247
535 H=H+1                                         :rem 198
540 NEXTJ                                         :rem 34
550 NEXTI                                         :rem 34
600 M=LEN(A$):M=((22-M)/2)                        :rem 90
605 GOSUB1000:PRINTL$TAB(M)A$                     :rem 237
620 GETE$:IF E$="X"THEN:PRINTK$:END               :rem 98
622 IFE$="V"THENGOSUB1000:PRINTL$;G$:GOTO710      :rem 193
625 IFE$=""THEN620                                :rem 221
627 IFE$="S"THENGOTO2000                          :rem 149
630 FORI=1TOA                                     :rem 27
640 IFE$=MID$(B$,I,1)THEN700                      :rem 28
650 NEXTI                                         :rem 35
700 GOSUB1000:IFI=<A THEN:PRINTL$;D$(I);" -";D(I) :rem 12
710 FORJ=1TO1000:NEXT                             :rem 20
720 GOTO600                                       :rem 104
1000 FORJ=7680TO7701                              :rem 118
1010 POKEJ,32                                     :rem 204
1020 NEXTJ                                        :rem 76
1030 RETURN                                       :rem 164
2000 PRINTK$"OPENING FILE"                        :rem 50
```

```
2005 REM SAVE GRAPH                          :rem 76
2010 OPEN1,1,1,A$                           :rem 113
2020 PRINT#1,A$                              :rem 54
2025 PRINT#1,G$                              :rem 65
2030 PRINT#1,A                               :rem 19
2040 PRINT#1,B                               :rem 21
2050 FORI=1TOA                               :rem 73
2060 PRINT#1,D$(I)                          :rem 215
2070 PRINT#1,C(I)                           :rem 179
2080 PRINT#1,D(I)                           :rem 181
2090 NEXTI                                   :rem 83
2100 PRINT"CLOSING FILE"                    :rem 195
2110 CLOSE 1                                :rem 107
```

Doug
Smoak

Rainbow Border

When I first started experimenting with changing the screen format (number of rows and columns) on the VIC, I wanted a way to use the screen border. It happened that I was also getting into machine language at the same time and was trying to duplicate things I had done in BASIC as exercises.

I decided to see what would happen when I changed decimal location 36879 (screen and border color register) as rapidly as possible in machine language. Wow! The entire VIC screen turned into fragments of different colors. It changed the screen color many times for each scan line on the TV. I decided that the routine needed a delay loop. After adding the loop and running the program again, I saw bars of colors scrolling very fast in the border. I had also changed the range of colors so that the screen color stayed the same. By fine-tuning the delay loop, I was able to broaden and stabilize the bars of color.

Color Scrolling

"Flag" was the first program I wrote to use this technique. It is very short and easy to type in, but as always be sure to get the DATA statements right. The machine language routine that does the work is represented by those numbers in the DATA statements, and if they are wrong, you could crash the program. So save the program to disk or tape before running it.

The next idea I got was to use multicolor characters on the screen, which I knew would be partly made up of border colors. This allows the scrolling colors to "show through" the characters where there is a bit that is in the border color. This is the technique used in "Rainbow Border." The PRINT statements in lines 30–45 can be replaced by whatever message you want. You can also stop the program and then type NEW. This leaves the machine language routine protected in the upper 256 bytes of memory on a VIC with the *Super Expander* cartridge, 3K RAM expansion, or standard memory. You can then do graphic work with the *Super Expander* or keyboard symbols and SYS 7424 to activate the color scroll.

This is especially good for *Super Expander* programs in GRAPHIC 1. What you do is develop a nice graphic design program and then add lines 10, 50, and 60 to call the routine to scroll the border colors.

One other thing to make this more flexible: The screen and border range of colors is set by the second and thirteenth DATA numbers. These should be changed in increments of 16 to change the screen color. Also make sure you put the SYS 7424 after the design part of the program since the machine language routine doesn't return to BASIC.

The third program, "S.X. Demo" is for the *Super Expander.* It is a demonstration of having the border colors "show through" the design. Note

that to use the border colors in the design, your design must be in GRAPHIC 1, and the first number after CIRCLE, DRAW, and so forth, must be 1. This is a short and easy program to type in, but again be careful with the DATA statements.

Program 1. Flag

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
20 PRINT"{CLR}{WHT}";:FORI=7424TO7424+32:READK:POKEI,K:NEXT
                                                    :rem 197
30 REM**TO DISABLE STOP KEY**:POKE 808,100          :rem 123
40 POKE56,29:POKE52,29                              :rem 252
50 FORT=1TO5                                        :rem 230
60 PRINT"{10 SPACES}";                              :rem 114
70 PRINT"**********";:NEXT                          :rem 144
80 SYS7424                                           :rem 56
90 DATA173,2,144,41,128,9,10,141,2,144,169,150,141,3,144,160,
   106,140                                           :rem 43
100 DATA15,144,162,137,202,208,253,136,192,104,208,243,76,15,
    29                                               :rem 74
```

Program 2. Rainbow Border

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}";:FORI=7424TO7424+17:READK:POKEI,K:NEXT:
                                                    :rem 252
20 POKE646,13:POKE56,29:POKE52,29:POKE36878,32      :rem 157
25 REM ***POKE 808,100 TO DISABLE STOP KEY***       :rem 153
30 PRINT"{CLR}{2 DOWN}{2 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}
   {5 SPACES}{2 +}"                                  :rem 39
31 PRINT"{2 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}{4 SPACES}{+}
   {2 SPACES}{+}"                                   :rem 115
32 PRINT"{2 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}
   "                                                :rem 206
33 PRINT"{2 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}
   "                                                :rem 207
34 PRINT"{2 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}{3 SPACES}{+}
   "                                                :rem 208
35 PRINT"{3 SPACES}{+} {+}{4 SPACES}{+}{3 SPACES}{+}
   {3 SPACES}"                                      :rem 209
36 PRINT"{3 SPACES}{+} {+}{4 SPACES}{+}{4 SPACES}{+}
   {2 SPACES}{+}"                                   :rem 120
37 PRINT"{4 SPACES}{+}{5 SPACES}{+}{5 SPACES}{2 +}{3 DOWN}"
                                                     :rem 6
38 PRINT"{5 SPACES}QQQ{5 SPACES}QQQ"                 :rem 34
39 PRINT"{4 SPACES}Q{3 SPACES}Q{3 SPACES}Q{3 SPACES}Q"
                                                    :rem 129
40 PRINT"{8 SPACES}Q{3 SPACES}Q{3 SPACES}Q"         :rem 168
```

```
41 PRINT"{8 SPACES}Q{3 SPACES}Q{3 SPACES}Q"          :rem 169
42 PRINT"{7 SPACES}Q{4 SPACES}Q{3 SPACES}Q"          :rem 170
43 PRINT"{6 SPACES}Q{5 SPACES}Q{3 SPACES}Q"          :rem 171
44 PRINT"{5 SPACES}Q{6 SPACES}Q{3 SPACES}Q"          :rem 172
45 PRINT"{4 SPACES}QQQQQ{4 SPACES}QQQ"               :rem 194
50 SYS7424                                           :rem 53
60 DATA160,111,140,15,144,162,136,202,208,253,136,192,103,208
   ,243,76,0,29                                      :rem 42
```

Program 3. S.X. Demo

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 GRAPHIC1:COLOR0,1,5,5                             :rem 152
20 FORX=0TO500STEP50                                 :rem 231
30 CIRCLE1,511,511,511-X,X                           :rem 152
40 NEXT                                              :rem 163
50 FORI=7424TO7424+17:READK:POKEI,K:NEXT             :rem 237
60 DATA160,15,140,15,144,162,136,202,208,253,136,192,7,208,24
   3,76,0,29                                         :rem 160
100 SYS7424                                          :rem 97
```

Brian Zupke

Music Maker

"Music Maker" will help you create music on your VIC. It requires at least 8K of memory expansion.

There are six parameters associated with making music. They are the four voices, the volume, and the duration. These six parameters make up a music set. By placing music sets one after another, you can create a composition of music. Each music set takes six bytes of memory; therefore, you can have a lot of music on an expanded VIC.

There are two programs used for creating, playing, and saving music. They are "Music Maker" and "Music Player." Music Maker allows you to create, edit, and play music sets. You can save music in a file and load it into memory for later use. Music Player contains the machine language routine and takes care of memory pointers. After you have created the music, you can combine this program with your own.

Music Maker

In the main menu, you have eight choices. You can Add Music Sets, Edit Music Sets, Save Music, Load Music, Turn Music On, Turn Music Off, Erase Music, or Quit Program.

Add Music Sets. The set number that you are adding is displayed. The first music set is set 0. Below the displayed set number, all the sound parameters are listed. These are voices 1–4, the loudness and duration of the set. After each question mark the previous value of that parameter is displayed. This makes entering the same values for the parameters easy. It also shows you what the previous sound set had. The four voices are in the form of *ON*, where *O* is the octave (1, 2, or 3) and *N* is the note. The note can be A–G plus sharps. The # symbol is used for sharps. No flats are allowed. If a voice is to be silent, then its value will be *NN*. All four voices start off with the value of *NN*.

The loudness can be from 0 to 15. A value of 0 would mean there would be no sound heard. This is one way to create a rest in the music. If a value greater than 15 is typed in, it is changed to 15; a value of less than 0 is changed to 0. The duration can be from 0 to 255. The number of seconds can be calculated by dividing this value by 60. A value of 0 would give the shortest duration (1/60 second), and 255 the longest (255/60, or approximately 4.25 seconds). When entering values, you must press the RETURN key on each line. The cursor keys should not be used. After values for all the parameters are entered, you are asked if you are finished adding music sets. If you wish to enter another music set, press the RETURN key; otherwise, type a Y to return to the main menu.

Edit Music Sets. The edit mode has its own menu displaying seven functions: View/Change a Set, Play a Set, Play Composition, Step Composition, Insert a Set, Delete a Set, and Quit Edit Mode.

The View/Change a Set function allows you to look at a music set and change its parameters if you wish. After you type in a number for an existing set, the values of the parameters associated with that set are displayed. They are in the same format as they are in the Add mode. Type Y to change something, and the cursor is moved up to the top of the list. The set is reentered, and the old set in memory is replaced with the new one. After the set is reentered, the screen is cleared, and you are asked for another set number. If you are through, type a negative number.

The Play a Set function will let you hear how any set sounds. When asked for a set number, enter the number you wish; a negative number will return control to the edit menu.

The Play Composition function plays all the music sets that you have created and displays the number of the set currently being played.

The Step Composition function is similar to the Play Composition function; however, after each set is played, you must press any key to hear the next set.

Use the Insert a Set function to insert a new set anywhere in the composition. Enter a valid set number. The new set will be inserted in the position of the number you typed. The set that is currently in that position will shift to follow the new set. Type in the values for the new set. If you are through, type a negative number.

Be careful using the Delete a Set function, for once a set is deleted, you cannot recall it. When you type in a set number, the values of the parameters of that set are displayed. Any sets that come after the one deleted are moved down to fill the gap. Once again, enter a negative number to return to the menu.

The Quit Edit Mode function returns control to the main menu.

Save Music. The music can be saved in a file on tape or disk. You are asked for the name of the file you wish to save the music under. If you don't wish to save the music, type STOP, and control is returned to the main menu. Otherwise, a file is opened under the designated name, and the music is saved. Control is returned to the menu after it is saved.

Disk users should change the following lines:

```
3400 OPEN 1,8,2,N$+",S,W":PRINT#1,SS
3520 OPEN 1,8,2,N$:INPUT#1,SS
```

Load Music. As with saving music, you are asked for the name of the file. Entering STOP returns control to the main menu. Type in the name of the file to load it. The proper file is then opened, and the set is loaded into memory.

Turn Music On. This changes the interrupt pointers to point to the music machine language routine. If you have any music sets in memory, they will be played after the time delay. After all the sets have been played, the silent period occurs again, and the music starts over. While the music is "on," you can add or edit the music. This allows you to hear the additions or changes as you make them.

Turn Music Off. This changes the interrupt pointers back to their original values and turns off the music.

Erase Music. If you want to start from scratch, use this function. It will erase any existing music sets in memory. You are asked if you are sure you want to do this. You must type YES to erase the music.

Quit Program. This stops the program, but you are first asked if you are sure you want to end it. You must type YES to stop the program. If you want to get back into the program, type GOTO 3610 and control will be returned to the main menu.

Music Player

The operation of this program is quite simple. It contains three machine language routines: the music routine, turn music off, and turn music on. If you use this routine in your program, turn the music on with SYS UN; to turn it off, type SYS OF.

When you turn the music on, the time delay starts and the music starts playing with the first set. When you turn it off, the machine language routine stops executing and the sound registers are cleared.

If you use this program as an introduction to your program, you'll need to know where the two machine language routines begin, since all variables are lost. Check the values of UN and OF before loading the second program.

Disk users should change line 620 to

620 OPEN 1,8,2,N$:INPUT#1,SS

Program 1. Music Maker

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1140 GOTO3900                                        :rem 203
1150 REM                                             :rem 171
1160 REM                                             :rem 172
1170 REM ****** SUBROUTINES ******                     :rem 8
1180 REM                                             :rem 174
1190 REM                                             :rem 175
1200 REM                                             :rem 167
1210 REM *** DISPLAY SOUND SET ***                    :rem 47
1220 PRINT"{DOWN}"                                   :rem 167
1230 FORX=1TO4:PRINT"VOICE #"X"? ";N$(X):NEXT       :rem 152
1240 PRINT"LOUDNESS?"V:PRINT"DURATION?"D             :rem 142
```

```
1250 RETURN                                                :rem 168
1260 REM                                                   :rem 173
1270 REM *** INPUT SOUND SET ***                           :rem 175
1280 REM                                                   :rem 175
1290 FORX=1TO4:PRINT"VOICE #"X;:INPUTN$(X):NEXT            :rem 229
1300 INPUT"LOUDNESS";V:INPUT"DURATION";D                   :rem 137
1310 RETURN                                                :rem 165
1320 REM                                                   :rem 170
1330 REM *** STORE SET IN MEMORY ***                       :rem 144
1340 REM                                                   :rem 172
1350 IFV>15THENV=15                                        :rem 122
1360 IFV<0THENV=0                                           :rem 13
1370 POKEBS+SN*6,V                                          :rem 61
1380 FORX=1TO4:O$=LEFT$(N$(X),1):N$=RIGHT$(N$(X),LEN(N$(X))-1
     )                                                      :rem 30
1390 O=VAL(O$):IFO=0ORO>3THENT=0:GOTO1430                   :rem 17
1400 FORY=1TO12:IFM$(Y)=N$THEN1420                           :rem 2
1410 NEXTY:T=0:GOTO1430                                    :rem 148
1420 T=V((O-1)*12+Y)                                        :rem 14
1430 POKEBS+SN*6+X,T:NEXTX                                 :rem 140
1440 IFD+1>255THEND=0                                      :rem 178
1450 IFD<1THEND=1                                          :rem 235
1460 POKEBS+SN*6+5,D+1                                     :rem 231
1470 RETURN                                                :rem 172
1480 REM                                                   :rem 177
1490 REM *** PLAY SET ***                                  :rem 208
1500 REM                                                   :rem 170
1510 P=PEEK(BS+SN*6):POKE36878,P:TM=TI                      :rem 53
1520 FORX=1TO4:P=PEEK(BS+SN*6+X):POKE36873+X,P:NEXT        :rem 185
1530 P=PEEK(BS+SN*6+5):TM=TM+P                              :rem 33
1535 IFTM>TITHEN1535                                       :rem 214
1540 RETURN                                                :rem 170
1550 REM                                                   :rem 175
1560 REM *** GET SET FROM MEMORY ***                       :rem 133
1570 REM                                                   :rem 177
1580 V=PEEK(BS+SN*6):FORX=1TO4:T=PEEK(BS+SN*6+X)           :rem 219
1590 IFT=0THENN$(X)="NN":GOTO1640                          :rem 196
1600 FORY=1TO36:IFT=V(Y)THEN1620                           :rem 211
1610 NEXTY:N$(X)="NN":GOTO1640                              :rem 16
1620 N$(X)=STR$(INT((Y-1)/12)+1):N$(X)=RIGHT$(N$(X),1)
                                                           :rem 190
1630 N$(X)=N$(X)+M$(Y-INT((Y-1)/12)*12)                     :rem 19
1640 NEXTX:D=PEEK(BS+SN*6+5)-1:IFD<0THEND=255              :rem 215
1650 RETURN                                                :rem 172
1660 REM                                                   :rem 177
1670 REM *** GET HIGH AND LOW BYTE OF ADDRESS **            :rem 24
1680 REM                                                   :rem 179
1690 H=INT(A/256):L=A-H*256:RETURN                         :rem 248
1700 REM                                                   :rem 172
1710 REM *** TURN OFF SOUND ***                             :rem 86
1720 REM                                                   :rem 174
```

```
1730 FORX=36878TO36877:POKEX,0:NEXT:RETURN                :rem 185
1740 REM                                                  :rem 176
1750 REM                                                  :rem 177
1760 REM ****** ADD SOUND SETS ******                      :rem 59
1770 REM                                                  :rem 179
1780 REM                                                  :rem 180
1790 PRINT"{CLR}{2 SPACES}ADD SOUND SETS{2 DOWN}"         :rem 232
1800 PRINT"SET NUMBER";SS"{DOWN}"                         :rem 133
1810 SN=SS                                                :rem 78
1820 REM                                                  :rem 175
1830 REM * PRINT SET *                                    :rem 125
1840 REM                                                  :rem 177
1850 GOSUB1220                                            :rem 19
1860 REM                                                  :rem 179
1870 REM * GET NEW SET *                                  :rem 190
1880 REM                                                  :rem 181
1890 PRINT"{HOME}{7 DOWN}";                               :rem 104
1900 GOSUB1290                                            :rem 22
1910 REM                                                  :rem 175
1920 REM * STORE NEW SET IN MEMORY *                      :rem 215
1930 REM                                                  :rem 177
1940 GOSUB1350                                            :rem 23
1950 SS=SS+1:FORX=0TO5:POKEBS+SS*6+X,255:NEXT:INPUT"DONE ADDI
     NG{2 SPACES}N{3 LEFT}";Y$                            :rem 29
1960 IFLEFT$(Y$,1)="N"THEN1790                            :rem 168
1970 GOTO3610                                             :rem 212
1980 REM                                                  :rem 182
1990 REM                                                  :rem 183
2000 REM ****** EDIT SOUND SETS ******                    :rem 140
2010 REM                                                  :rem 167
2020 REM                                                  :rem 168
2030 PRINT"{CLR}{3 SPACES}EDIT SOUND SETS{2 DOWN}"        :rem 57
2040 PRINT" V-VIEW/CHANGE A SET{DOWN}":PRINT" P-PLAY A SET
     {DOWN}"                                              :rem 100
2050 PRINT" C-PLAY COMPOSITION{DOWN}":PRINT" S-STEP COMPOSITI
     ON{DOWN}"                                            :rem 207
2060 PRINT" I-INSERT A SET{DOWN}":PRINT" D-DELETE A SET{DOWN}
     "                                                    :rem 143
2070 PRINT" Q-QUIT EDIT MODE{2 DOWN}":INPUT"{4 SPACES}CHOICE"
     ;Y$:Y$=LEFT$(Y$,1)                                   :rem 167
2080 IFY$="V"THEN2190                                     :rem 168
2090 IFY$="P"THEN2440                                     :rem 161
2100 IFY$="C"THEN2590                                     :rem 146
2110 IFY$="S"THEN2740                                     :rem 160
2120 IFY$="I"THEN2920                                     :rem 151
2130 IFY$="D"THEN3140                                     :rem 142
2140 IFY$="Q"THEN3610                                     :rem 158
2150 GOTO2030                                             :rem 198
2160 REM                                                  :rem 173
2170 REM *** VIEW SETS ***                                :rem 36
2180 REM                                                  :rem 175
```

```
2190 PRINT"{CLR}{2 SPACES}VIEW/CHANGE SETS{2 DOWN}":INPUT"SET
     NUMBER";SN:PRINT"{DOWN}"                          :rem 92
2200 IFSN<0THEN2030                                    :rem 84
2210 IFSN>=SSTHEN2190                                  :rem 17
2220 REM                                              :rem 170
2230 REM * GET SET FROM MEMORY *                      :rem 216
2240 REM                                              :rem 172
2250 GOSUB1580                                         :rem 23
2260 REM                                              :rem 174
2270 REM * DISPLAY SET *                                :rem 5
2280 REM                                              :rem 176
2290 GOSUB1220                                         :rem 18
2300 INPUT"{DOWN}CHANGE";Y$:IFLEFT$(Y$,1)<>"Y"THEN2190:rem 93
2310 REM                                              :rem 170
2320 REM * GET NEW SET *                              :rem 181
2330 REM                                              :rem 172
2340 PRINT"{HOME}{8 DOWN}";                           :rem 112
2350 GOSUB1290                                         :rem 22
2360 REM                                              :rem 175
2370 REM * STORE SET IN MEMORY *                      :rem 237
2380 REM                                              :rem 177
2390 GOSUB1350                                         :rem 23
2400 GOTO2190                                         :rem 203
2410 REM                                              :rem 171
2420 REM *** PLAY A SET ***                            :rem 11
2430 REM                                              :rem 173
2440 PRINT"{CLR}{5 SPACES}PLAY A SET{2 DOWN}"         :rem 179
2450 INPUT"SET NUMBER";SN                              :rem 48
2460 IFSN<0THEN2030                                    :rem 92
2470 IFSN>=SSTHEN2440                                  :rem 23
2480 REM                                              :rem 178
2490 REM * PLAY SET *                                  :rem 41
2500 REM                                              :rem 171
2510 GOSUB1510                                         :rem 15
2520 REM                                              :rem 173
2530 REM *** TURN OFF SOUND ***                        :rem 87
2540 REM                                              :rem 175
2550 GOSUB1730:GOTO2440                                :rem 84
2560 REM                                              :rem 177
2570 REM *** PLAY COMPOSITION ***                      :rem 56
2580 REM                                              :rem 179
2590 PRINT"{CLR}{2 SPACES}PLAY COMPOSITION"           :rem 190
2600 FORX=1TO1000:NEXT                                 :rem 82
2610 FORSN=0TOSS-1:PRINT"{HOME}{6 DOWN}SET NUMBER";SN:rem 122
2620 REM                                              :rem 174
2630 REM * PLAY SET *                                  :rem 37
2640 REM                                              :rem 176
2650 GOSUB1510                                         :rem 20
2660 REM                                              :rem 178
2670 REM *** TURN OFF SOUND ***                        :rem 92
2680 REM                                              :rem 180
```

```
2690 NEXT:GOSUB1730                                        :rem 149
2700 FORX=1TO1000:NEXT:GOTO2030                            :rem 139
2710 REM                                                   :rem 174
2720 REM *** STEP COMPOSITION ***                           :rem 59
2730 REM                                                   :rem 176
2740 PRINT"{CLR}{3 SPACES}STEP COMPOSITION{2 DOWN}"        :rem 227
2750 PRINT"PRESS ANY KEY TO HEAR{DOWN}":PRINT"THE NEXT SET"
                                                           :rem 232
2760 FORSN=0TOSS-1                                         :rem 107
2770 GETA$:IFA$=""THEN2770                                 :rem 195
2780 PRINT"{HOME}{8 DOWN}SET NUMBER";SN                    :rem 206
2790 REM                                                   :rem 182
2800 REM * PLAY SET *                                       :rem 36
2810 REM                                                   :rem 175
2820 GOSUB1510                                              :rem 19
2830 NEXT                                                   :rem 12
2840 GETA$:IFA$=""THEN2840                                 :rem 191
2850 REM                                                   :rem 179
2860 REM *** TURN OFF SOUND ***                             :rem 93
2870 REM                                                   :rem 181
2880 GOSUB1730:GOTO2030                                     :rem 85
2890 REM                                                   :rem 183
2900 REM *** INSERT A SET ***                              :rem 173
2910 REM                                                   :rem 176
2920 PRINT"{CLR}{4 SPACES}INSERT A SET{2 DOWN}"             :rem 85
2930 INPUT"NEW SET NUMBER";SN:IFSN<0THEN2030               :rem 231
2940 IFSN>=SSTHEN2920                                       :rem 28
2950 REM                                                   :rem 180
2960 REM * DISPLAY SET *                                    :rem 11
2970 REM                                                   :rem 182
2980 GOSUB1220                                              :rem 24
2990 REM                                                   :rem 184
3000 REM * GET NEW SET *                                   :rem 177
3010 REM                                                   :rem 168
3020 PRINT"{HOME}{6 DOWN}";                                 :rem 74
3030 GOSUB1290                                              :rem 18
3040 REM                                                   :rem 171
3050 REM *** SET PARAMETERS FOR ML ***                       :rem 8
3060 REM                                                   :rem 173
3070 A=BS+SS*6+5:GOSUB1690:POKE830,L:POKE831,H             :rem 237
3075 A=BS+SN*6:GOSUB1690:POKE828,L:POKE829,H               :rem 155
3080 SYS UP                                                :rem 111
3081 REM                                                   :rem 176
3082 REM *** STORE NEW SET INTO MEMORY ***                  :rem 35
3083 REM                                                   :rem 178
3085 GOSUB1350                                              :rem 25
3090 SS=SS+1                                               :rem 177
3100 PRINT"NEW SET INSERTED":FORX=1TO1000:NEXT:GOTO2920
                                                           :rem 205
3110 REM                                                   :rem 169
```

```
3120 REM *** DELETE A SET ***                                  :rem 134
3130 REM                                                       :rem 171
3140 PRINT"{CLR} DELETE A SET{2 DOWN}":INPUT"SET NUMBER";SN
                                                               :rem 205
3150 IFSN<0THEN2030                                             :rem 89
3160 IFSN>=SSTHEN3140                                           :rem 18
3170 REM                                                       :rem 175
3180 REM * GET SET FROM MEMORY *                               :rem 221
3190 REM                                                       :rem 177
3200 GOSUB1580                                                  :rem 19
3210 REM                                                       :rem 170
3220 REM * DISPLAY SET *                                         :rem 1
3230 REM                                                       :rem 172
3240 GOSUB1220                                                  :rem 14
3250 INPUT"OK TO DELETE";Y$:Y$=LEFT$(Y$,1)                     :rem 180
3260 IFY$<>"Y"THEN3140                                         :rem 229
3270 REM                                                       :rem 176
3280 REM *** SET PARAMETERS FOR ML ***                          :rem 13
3290 REM                                                       :rem 178
3300 A=BS+SN*6:GOSUB1690:POKE828,L:POKE829,H                   :rem 146
3305 A=BS+SS*6+5:GOSUB1690:POKE830,L:POKE831,H                 :rem 238
3310 SYS DOWN :PRINT"SET DELETED":SS=SS-1                       :rem 13
3320 FORX=1TO1000:NEXT:GOTO3140                                :rem 141
3330 REM                                                       :rem 173
3340 REM                                                       :rem 174
3350 REM ****** SAVE MUSIC ON TAPE ******                       :rem 30
3360 REM                                                       :rem 176
3370 REM                                                       :rem 177
3380 PRINT"{CLR}{3 SPACES}SAVE MUSIC {2 DOWN}":PRINT"NAME OF
     {SPACE}FILE:"                                              :rem 31
3390 INPUTN$:IFN$="STOP"THEN3610                               :rem 204
3400 OPEN1,1,1,N$:PRINT#1,SS                                   :rem 111
3410 FORX=0TOSS:FORY=0TO5                                      :rem 124
3420 P=PEEK(BS+X*6+Y)                                          :rem 200
3430 PRINT#1,P                                                  :rem 39
3440 NEXTY,X:CLOSE1:GOTO3610                                     :rem 5
3450 REM                                                       :rem 176
3460 REM                                                       :rem 177
3470 REM ****** LOAD MUSIC FROM TAPE ******                    :rem 169
3480 REM                                                       :rem 179
3490 REM                                                       :rem 180
3500 PRINT"{CLR}{2 SPACES}LOAD MUSIC{2 DOWN}":PRINT"NAME OF F
     ILE:"                                                      :rem 10
3510 INPUTN$:IFN$="STOP"THEN3610                               :rem 198
3520 OPEN1,1,0,N$:INPUT#1,SS                                   :rem 116
3530 FORX=0TOSS:FORY=0TO5                                      :rem 127
3540 INPUT#1,P:POKEBS+X*6+Y,P                                   :rem 13
3550 NEXTY,X:CLOSE1:GOTO3610                                     :rem 7
3560 REM                                                       :rem 178
3570 REM                                                       :rem 179
3580 REM ****** MAIN MENU ******                                  :rem 6
```

```
3590 REM                                                    :rem 181
3600 REM                                                    :rem 173
3610 PRINT"{CLR}{3 SPACES}MUSIC MAKER{DOWN}":PRINT"{6 SPACES}
     MENU{DOWN}"                                            :rem 129
3620 PRINT" A-ADD MUSIC SETS{DOWN}":PRINT" E-EDIT MUSIC SETS
     {DOWN}"                                                 :rem 24
3630 PRINT" S-SAVE MUSIC{DOWN}":PRINT" L-LOAD MUSIC{DOWN}"
                                                             :rem 20
3640 PRINT" O-TURN MUSIC ON{DOWN}":PRINT" F-TURN MUSIC OFF
     {DOWN}"                                                :rem 198
3650 PRINT" C-ERASE MUSIC{DOWN}"                             :rem 17
3660 PRINT" Q-QUIT PROGRAM{DOWN}"                           :rem 138
3670 INPUT"{3 SPACES}CHOICE";C$:C$=LEFT$(C$,1)               :rem 51
3680 IFC$="A"THEN1790                                       :rem 137
3690 IFC$="E"THEN2030                                       :rem 130
3700 IFC$="S"THEN3380                                       :rem 145
3710 IFC$="L"THEN3500                                       :rem 133
3720 IFC$="O"THEN SYS UN :GOTO3610                          :rem 160
3730 IFC$="F"THENSYS OF :GOTO3610                           :rem 138
3740 IFC$="C"THEN3790                                       :rem 138
3750 IFC$="Q"THEN3770                                       :rem 151
3760 GOTO3610                                               :rem 211
3770 INPUT"ARE YOU SURE";Y$:IFY$="YES"THENEND                :rem 47
3780 GOTO3610                                               :rem 213
3790 INPUT"ARE YOU SURE";Y$:IFY$="YES"THEN3810               :rem 38
3800 GOTO3610                                               :rem 206
3810 SS=0:FORX=BSTOBS+5:POKEX,255:NEXT:GOTO3610                :rem 1
3820 REM                                                    :rem 177
3830 REM                                                    :rem 178
3840 REM ****** PROGRAM START ******                          :rem 81
3850 REM                                                    :rem 180
3860 REM                                                    :rem 181
3870 REM                                                    :rem 182
3880 REM * STORE TOP OF MEMORY VALUE *                      :rem 118
3890 REM                                                    :rem 184
3900 X=PEEK(55):POKE900,X:X=PEEK(56):POKE901,X              :rem 254
3910 REM                                                    :rem 177
3920 REM * SET NEW TOP OF MEMORY *                           :rem 61
3930 REM                                                    :rem 179
3940 BS=12800:A=BS:GOSUB1690:BH=H:BL=L                       :rem 14
3950 POKE55,BL:POKE51,BL:POKE56,BH:POKE52,BH:CLR            :rem 216
3960 BS=12800:MS=PEEK(900)+PEEK(901)*256-236                :rem 143
3970 DOWN=MS+103:UP=MS+146:UN=MS+189:OF=MS+212:DU=60        :rem 146
3980 FORX=BSTOBS+5:POKEX,255:NEXT:BS=BS-1                   :rem 126
3990 A=BS:GOSUB1690:SH=H:SL=L:A=MS:GOSUB1690:MH=H:ML=L:BS=BS+
     1                                                      :rem 189
4000 REM                                                    :rem 168
4010 REM * LOAD MUSIC TABLES *                               :rem 89
4020 REM                                                    :rem 170
4030 DIMV(36),M$(12):FORX=1TO36:READV(X):NEXT               :rem 195
4040 FORX=1TO12:READM$(X):NEXT                              :rem 100
```

```
4050 REM                                                  :rem 173
4060 REM * STORE MACHINE LANGUAGE PROGRAM IN MEMORY * :rem 80
4070 REM                                                  :rem 175
4080 FORX=MSTOMS+232:READA$:IFLEFT$(A$,1)<"A"THENPOKEX,VAL(A$
     ):GOTO 4130                                           :rem 74
4090 IFA$="SH"THENPOKEX,SH                                 :rem 90
4100 IFA$="SL"THENPOKEX,SL                                 :rem 90
4110 IFA$="ML"THENPOKEX,ML                                 :rem 79
4120 IFA$="MH"THENPOKEX,MH                                 :rem 72
4125 IFA$="DU"THENPOKEX,DU                                 :rem 85
4130 NEXTX                                                 :rem 95
4140 FORX=1TO4:N$(X)="NN":NEXT                             :rem 56
4150 GOTO3610                                             :rem 205
4160 REM                                                  :rem 175
4170 REM * DATA FOR MUSIC VALUES AND NOTES *              :rem 178
4180 REM                                                  :rem 177
4190 DATA135,143,147,151,159,163,167,175,179,183,187,191,195,
     199,201,203,207,209,212                              :rem 131
4200 DATA215,217,219,221,223,225,227,228,229,231,232,233,235,
     236,237,238,239                                      :rem 216
4210 DATAC,C#,D,D#,E,F,F#,G,G#,A,A#,B                     :rem 165
4220 REM                                                  :rem 172
4230 REM * MACHINE LANGUAGE PROGRAM *                      :rem 82
4240 REM                                                  :rem 174
4250 DATA198,253,208,36,160,0,230,251,208,2,230,252,177,251
                                                          :rem 133
4260 DATA201,255,208,25,162,4,169,0,157,10,144,202,16,250
                                                           :rem 24
4270 DATA169,SH,133,252,169,SL,133,251,169,DU             :rem 133
4280 DATA133,253,76,191,234,141,14,144,230,251,208,2,230,252
                                                          :rem 177
4290 DATA177,251,141,10,144,230,251,208,2,230,252,177,251,141
                                                          :rem 224
4300 DATA11,144,230,251,208,2,230,252,177,251,141,12,144,230
                                                          :rem 157
4310 DATA251,208,2,230,252,177,251,141,13,144,230,251,208,2
                                                          :rem 115
4320 DATA230,252,177,251,133,253,76,191,234               :rem 122
4330 REM                                                  :rem 174
4340 REM * MOVE MUSIC SETS DOWN ROUTINE *                  :rem 88
4350 REM                                                  :rem 176
4360 DATA173,60,3,133,97,173,61,3,133,98,160,6,177,97:rem 119
4370 DATA160,0,145,97,230,97,208,2,230,98,165,98,205,63
                                                          :rem 217
4380 DATA3,48,235,240,233,165,97,205,62,3,48,226,240,224,96
                                                          :rem 153
4390 REM                                                  :rem 180
4400 REM * MOVE MUSIC SETS UP *                           :rem 156
4410 REM                                                  :rem 173
4420 DATA173,62,3,133,97,173,63,3,133,98,160,0,177,97:rem 114
```

```
4425 DATA160,6,145,97,165,98,205,61,3,208,8,165,97,205
                                                    :rem 175
4430 DATA60,3,208,1,96,198,97,16,2,198,98,208,225,240:rem 123
4440 DATA223                                        :rem 125
4450 REM                                            :rem 177
4460 REM * TURN MUSIC ON *                          :rem 109
4470 REM                                            :rem 179
4475 DATA169,DU,133,253,169,SL,133,251,169,SH,133,252 :rem 21
4480 DATA169,ML,141,20,3,169,MH,141,21,3,96         :rem 214
4490 REM                                            :rem 181
4500 REM * TURN MUSIC OFF *                         :rem 166
4510 REM                                            :rem 174
4520 DATA169,191,141,20,3,169,234,141,21,3           :rem 60
4530 DATA162,4,169,0,157,10,144,202,16,250,96       :rem 211
```

Program 2. Music Player

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
100 REM                                             :rem 117
110 REM                                             :rem 118
120 REM *****{2 SPACES}MUSIC PLAYER{2 SPACES}*****  :rem 105
130 REM                                             :rem 120
140 REM                                             :rem 121
150 REM * STORE TOP OF MEMORY VALUE *                :rem 57
160 REM                                             :rem 123
170 X=PEEK(55):POKE900,X:X=PEEK(56):POKE901,X       :rem 202
180 REM                                             :rem 125
190 REM * SET NEW TOP OF MEMORY *                     :rem 9
200 REM                                             :rem 118
210 BS=12800:BH=INT(BS/256):BL=BS-BH*256             :rem 22
220 POKE55,BL:POKE51,BL:POKE56,BH:POKE52,BH:CLR     :rem 155
230 BS=12800:MS=PEEK(900)+PEEK(901)*256-150          :rem 77
240 UN=MS+103:OF=MS+126:DU=60                       :rem 187
250 BS=BS-1                                          :rem 92
260 SH=INT(BS/256):SL=BS-SH*256:MH=INT(MS/256):ML=MS-MH*256:B
    S=BS+1                                           :rem 49
270 REM                                             :rem 125
280 REM * STORE MACHINE LANGUAGE PROGRAM IN MEMORY *  :rem 32
290 REM                                             :rem 127
300 FORX=MSTOMS+146:READA$:IFLEFT$(A$,1)<"A"THENPOKEX,VAL(A$)
    :GOTO 360                                       :rem 230
310 IFA$="SH"THENPOKEX,SH                            :rem 33
320 IFA$="SL"THENPOKEX,SL                            :rem 42
330 IFA$="ML"THENPOKEX,ML                            :rem 31
340 IFA$="MH"THENPOKEX,MH                            :rem 24
350 IFA$="DU"THENPOKEX,DU                            :rem 33
360 NEXTX                                            :rem 48
370 REM                                             :rem 126
380 REM * MACHINE LANGUAGE PROGRAM *                 :rem 36
```

```
390 REM                                                    :rem 128
400 DATA198,253,208,36,160,0,230,251,208,2,230,252,177,251
                                                            :rem 78
410 DATA201,255,208,25,162,4,169,0,157,10,144,202,16,250
                                                           :rem 225
420 DATA169,SH,133,252,169,SL,133,251,169,DU                :rem 78
430 DATA133,253,76,191,234,141,14,144,230,251,208,2,230,252
                                                           :rem 122
440 DATA177,251,141,10,144,230,251,208,2,230,252,177,251,141
                                                           :rem 169
450 DATA11,144,230,251,208,2,230,252,177,251,141,12,144,230
                                                           :rem 111
460 DATA251,208,2,230,252,177,251,141,13,144,230,251,208,2
                                                            :rem 69
470 DATA230,252,177,251,133,253,76,191,234                  :rem 76
480 REM                                                    :rem 128
490 REM * TURN MUSIC ON *                                   :rem 60
500 REM                                                    :rem 121
510 DATA169,DU,133,253,169,SL,133,251,169,SH,133,252       :rem 215
520 DATA169,ML,141,20,3,169,MH,141,21,3,96                 :rem 157
530 REM                                                    :rem 124
540 REM * TURN MUSIC OFF *                                 :rem 118
550 REM                                                    :rem 126
560 DATA169,191,141,20,3,169,234,141,21,3                   :rem 12
570 DATA162,4,169,0,157,10,144,202,16,250,96               :rem 163
600 REM                                                    :rem 122
603 REM * LOAD MUSIC FROM FILE *                           :rem 198
605 REM                                                    :rem 127
610 INPUT"FILE NAME";N$                                     :rem 89
620 OPEN1,1,0,N$:INPUT#1,SS                                 :rem 66
630 FORX=0TOSS:FORY=0TO5                                    :rem 77
640 INPUT#1,P:POKEBS+X*6+Y,P                               :rem 219
650 NEXTY,X:CLOSE1                                         :rem 152
670 REM                                                    :rem 129
680 REM * PLACE MUSIC DATA HERE *                          :rem 250
690 REM                                                    :rem 131
```

Chapter Four

Applications

F. Wendell Johnson

Write-On

This is a word processor written in BASIC for the VIC with a Datassette and a serial printer (directions for adapting the program to print on a Commodore printer are included below). It should adapt easily to almost any home computer since the only command unique to VIC-20 is the "Bell" in lines 90 and 91. This program is not well-suited to an unexpanded VIC because available memory accommodates only about a dozen lines of text.

In addition to providing for typing text into memory and sending it to a serial printer, the program allows the user to relocate blocks of text, edit lines, and have the computer find and change specific words or phrases.

After the initial LOAD and RUN commands, the program goes first to a SET MARGIN routine to make sure the user established a line length. When this has been done, a one-line menu appears:

A, C, D, E, F, I, L, M, P, Q, S

Each letter represents one of the program functions: ADD, CHANGE, DELETE, EDIT, and so on. A function is selected by pressing the key corresponding to the desired function. When this is done, the name of the function appears and the program awaits data from the keyboard relating to the function.

ADD. Used for initial entry of text or for adding lines to text already entered. When A is selected from the menu, the program is immediately ready to accept the next line of text. The computer should be in the uppercase and lowercase mode if you want the output to be in that form. The lines will be numbered sequentially as they are entered, and a bell will signal when only five spaces remain on a line. More than one line on the screen may be needed to display one line of text.

The @ key provides a five-space tab whenever it is used, and the RETURN key terminates a line. If the RETURN is used on an empty line, then the ADD function ceases and the menu reappears. The DELETE key functions normally in the ADD mode, but the CLR/HOME, INSERT, and cursor movement keys should not be used. An interesting peculiarity occurs when quotation marks are used. Any letter enclosed in quotes will be preceded by two graphics characters. These characters are not included in the line length, and they will not be listed or printed. They may be ignored.

CHANGE. Directs the program to search through a specified range of lines of text and change *wrong word* to *right one* wherever it occurs. The range must be specified in this format: A, B RETURN, where A is the lowest numbered line of the range and B is the highest numbered line. The desired change must be in this format: / *wrong word* / *right one* / RETURN. *Wrong word* can be a letter, a word, or a phrase, and *right one* can be any of these

or nothing (for example, / you know / / RETURN would result in the deletion of the phrase *you know* wherever it appeared). It may take several seconds for this operation to be completed, but the menu will reappear when it has finished. It will be necessary to LIST the text in order to see the result of CHANGE. Almost any letter or symbol may be used in the CHANGE format in place of the /. The only requirement is that the character used must not appear in either the field to be changed or the field which replaces it.

DELETE. Causes a range of lines of text to be deleted. The range must be specified in this format: A, B RETURN, where A is the lowest numbered line to be deleted and B is the highest numbered line. If only one line is to be deleted, then A and B must be the same value. Failure to input two line numbers will result in another prompt (with two question marks). Input the second number and RETURN to proceed. The menu will reappear when DELETE is complete.

EDIT. Recalls a specific line of text for editing. The number of the line must be specified in this format: A RETURN. This results in the display of the line to be edited along with a workspace for the editing. When the cursor-right key is pressed, the line to be edited is copied into the workspace one character at a time. A character to be changed is typed in from the keyboard at the appropriate place on the line. A character to be deleted is copied, then deleted using the INST/DEL key. The line may then be completed by continuing to copy. A character (or a space) may be inserted by copying over to the appropriate place, pressing the up arrow (not cursor up), and then typing the character to be inserted. The up arrow must be used before each character to be inserted. RETURN terminates both the line and the EDIT function bringing back the menu.

FILER. Saves text or loads text from tape or disk into the program. LOAD or SAVE is specified by keying L or S. A filename must then be specified with a name which does not exceed 12 characters. When saving is complete, the menu returns. Loaded text always starts with line 1. Disk users should change OPEN1,1,0,L$ in line 41 to OPEN1,8,2,L$. Also change OPEN1,1,1,L$ in line 43 to OPEN1,8,2,L$+",S,W".

INSERT. Provides for the insertion of one or more lines of new text between any two lines of existing text. The number of the line of existing text to be preceded by the new text is specified in this format: A RETURN. This results in the creation of a work area with a new line number. When the new line has been typed in, press the RETURN key to prepare for another new line. Pressing RETURN on an empty line terminates INSERT and restores the menu.

LIST. Displays the text on the screen. A range of lines to be displayed may be specified in this format: A, B RETURN, where A is the lowest numbered line of the range, and B is the highest numbered line. If you press RETURN without specifying a range, then all lines will scroll by and the menu returns.

MOVE. Relocates a block of text. The range of lines in the block and the destination must be specified in this format: A, B, C RETURN, where A is the lowest numbered line of the range, B is the highest numbered line, and C is the line of text which will be preceded by the block when the move is complete. If the block is to be at the beginning of the text, C is specified as 1. If the block is to be at the end of the text, C is specified as one number higher than the last numbered line of text. The menu returns when the move is complete.

PRINT. Sends a range of lines of text to the printer without printing the line numbers. The range of lines may be specified in this format: A, B, RETURN, where A is the lowest numbered line in the range and B is the highest numbered line. If RETURN is pressed without specifying a range, then all lines will be printed. The menu returns when printing is complete.

QUIT. Terminates "Write-On" and displays the space remaining in memory for lines of text. This is the only exception to the general rule that pressing RETURN restores the menu. The program may be restarted by the command CONT or RUN. The RUN command will *erase all text* in memory.

SET MARGIN. Provides for specifying line length in this format: A RETURN. Margin will be evenly divided on either side of the page. Pressing RETURN without specifying the line length will default to a line length setting of 60 characters. The menu returns when RETURN is pressed.

Tailoring the Program

The program may be readily modified to meet specific needs. Here are the locations of the various functions:

1	Initialization
2–4	Input Subroutines
6–8	Menu
10, 11	ADD
15–23	CHANGE
25, 26	DELETE
30–35	EDIT
40–49	FILER
55–57	INSERT
60	LIST
65–69	MOVE
70–76	PRINT
80	QUIT
85	SET MARGIN
90–95	Build Lines

I designed the program to operate with my IDS Bright-Writer ASCII-compatible printer. If you have a Commodore printer, you may simplify the PRINT functions by deleting lines 71, 73, 74, changing L$ to T$(I) in line 75,

and shortening line 70 past the FOR I = L to H statement. You may also delete the OPEN 9,2,3 CHR$(6) from line 1, add OPEN 9,4 to line 70, and add CLOSE 9 to line 76 before GOTO 6.

Lines 1–11, 60, and 90–95 are the only lines absolutely necessary for the program to run. The others are options which you may include as you need them, modify as necessary, or replace with more powerful functions which you may design.

Write-On

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

Some lines of this program require keywords to be abbreviated so that they will not exceed the four-screen-line limit. See Appendix B.

```
1 OPEN9,2,3,CHR$(6):DIMT$(30):D$=CHR$(20):R$=CHR$(18):O$=CHR$
  (146):Z=1:M=60:GOTO85                                  :rem 107
2 GETA$:ON-(A$="")GOTO2:RETURN                            :rem 55
3 L=0:H=0:INPUTL,H:ON-(L+H=0)GOTO6:RETURN                :rem 111
4 L=0:INPUTL:ON-(L<1ORL>Z)GOTO6:RETURN                    :rem 31
6 P=36875:E$=CHR$(255):X$=CHR$(157):PRINTR$"A,C,D,E,F,I,L,M,P
  ,Q,S"O$:GOSUB2                                          :rem 65
7 J=0:FORI=1TO11:IFA$=MID$("ACDEFILMPQS",I,1)THENJ=I:I=11
                                                          :rem 229
8 NEXTI:ON-(J=0)GOTO6:ONJGOTO10,15,25,30,40,55,60,65,70,80,85
                                                          :rem 185
10 PRINTK$;R$"ADD"O$:PRINT                                :rem 85
11 PRINTZ">";:GOSUB90:ON-(LEN(L$)=0)GOTO6:T$(Z)=L$:Z=Z+1:GOTO
   11                                                     :rem 96
15 PRINTR$"CHANGE"O$;:GOSUB3:GOSUB90:N=LEN(L$):IFN<4THEN15
                                                          :rem 146
16 A$=LEFT$(L$,1):IFRIGHT$(L$,1)<>A$THEN15                :rem 115
17 J=0:FORI=2TON-1:IFMID$(L$,I,1)=A$THENJ=I               :rem 66
18 NEXTI:ON-(J=0ORJ=2)GOTO15:A$=MID$(L$,2,J-2):IFJ+1=NTHENS$=
   "":GOTO20                                              :rem 186
19 S$=MID$(L$,J+1,N-J-1)                                  :rem 254
20 A=LEN(A$):FORI=LTOH:B=LEN(T$(I)):L$="":S=1:FORJ=1TOB-A+1:I
   FMID$(T$(I),J,A)<>A$THEN22                             :rem 123
21 L$=L$+MID$(T$(I),S,J-S)+S$:S=J+A:J=S-1                 :rem 21
22 NEXTJ:IFS<>1THENL$=L$+RIGHT$(T$(I),B-S+1):T$(I)=L$:rem 115
23 NEXTI:GOTO6                                            :rem 150
25 PRINTR$"DELETE"O$;:GOSUB3:IFH=Z-1THENZ=L:GOTO6         :rem 189
26 J=H-L+1:FORI=LTOZ-J-1:T$(I)=T$(I+J):NEXTI:Z=Z-J:GOTO6
                                                          :rem 238
30 PRINTR$"EDIT"O$;:GOSUB4:PRINT:PRINTL">"T$(L):PRINTL">";:L$
   ="":I=1                                                :rem 201
31 GOSUB2:IFA$=C$THENT$(L)=L$:PRINT:GOTO6                 :rem 73
32 ON-(A$<>CHR$(29))GOTO33:A$=MID$(T$(L),I,1):GOTO35 :rem 168
33 IFA$=D$THENL$=LEFT$(L$,LEN(L$)-1):PRINTA$;:GOTO31      :rem 83
34 IFA$=CHR$(94)THEN:I=I-1:GOTO31                         :rem 196
35 I=I+1:PRINTA$;:L$=L$+A$:GOTO31                         :rem 184
```

```
40 PRINTK$;R$"FILER"O$" L OR S ?";:GOSUB2:PRINTA$:S$=A$:IFS$<
   >"L"ANDS$<>"S"THEN6                                  :rem 173
41 PRINT"NAME?";:GOSUB90:N=LEN(L$):ON-(N<1ORN>12)GOTO41:IFS$=
   "L"THENOPEN1,1,0,L$:Z=0:GOTO46                       :rem 105
42 IFZ=1THENPRINT"NO FILE":GOTO6                         :rem 35
43 OPEN1,1,1,L$:FORI=1TOZ-1:FORJ=1TOLEN(T$(I)):PRINT#1,MID$(T
   $(I),J,1);:NEXTJ                                     :rem 138
44 PRINT#1,E$;:NEXTI:CLOSE1:PRINTL$" SAVED":GOTO6        :rem 83
46 Z=Z+1:T$(Z)=""                                       :rem 149
47 GET#1,A$:ON-(ST=64)GOTO49:IFST<>0THENPRINT"LOAD ERROR"GOTO
   6                                                    :rem 172
48 ON-(A$=E$)GOTO46:T$(Z)=T$(Z)+A$:GOTO47                :rem 96
49 CLOSE1:PRINTL$" LOADED":Z=Z+1:GOTO6                  :rem 104
55 PRINTR$"INSERT"O$;:GOSUB4:PRINTL">";                 :rem 242
56 GOSUB90:IFLEN(L$)<1THEN6                             :rem 143
57 Z=Z+1:FORI=ZTOLSTEP-1:T$(I)=T$(I-1):NEXTI:T$(L)=L$:L=L+1:P
   RINTL">";:GOTO56                                      :rem 19
60 L=1:H=Z-1:PRINTK$;R$"LIST"O$;:INPUTL,H:FORI=LTOH:PRINTI">"
   T$(I):NEXTI:GOTO6                                     :rem 35
65 PRINTR$"MOVE"O$;:INPUTL,H,A:B=A-1:J=H-L+1:IF(FRE(1)/M<J)TH
   ENPRINT"NO ROOM":GOTO6                               :rem 194
66 N=Z-1:FORI=1TOJ:T$(N+I)=T$(L-1+I):NEXT               :rem 214
67 FORI=LTON-J:T$(I)=T$(I+J):NEXT:ON-(B=N)GOTO69:IFB>LTHENB=B
   -J                                                   :rem 221
68 FORI=NTOA+JSTEP-1:T$(I)=T$(I-J):NEXT:FORI=1TOJ:T$(B+I)=T$(
   N+I):NEXT:GOTO6                                       :rem 15
69 FORI=1TOJ:T$(N-J+I)=T$(N+I):NEXT:GOTO6                :rem 32
70 L=1:H=Z-1:PRINTR$"PRINT"O$;:INPUTL,H:FORI=LTOH:L$="":FORJ=
   1TO99:A$=MID$(T$(I),J,1)                             :rem 229
71 ON-(J>LEN(T$(I)))GOTO75:N=ASC(A$):IF64<NANDN<91THENA$=CHR$
   (N+32):GOTO74                                        :rem 167
73 IF192<NANDN<219THENA$=CHR$(N-128)                     :rem 81
74 L$=L$+A$:NEXTJ                                       :rem 219
75 CMD9:PRINTSPC(E)L$:NEXTI                             :rem 238
76 ON-(ST=0AND(PEEK(37151)AND64)=1)GOTO76:PRINT#9:GOTO6
                                                         :rem 84
80 PRINTK$R$"GOODBY"O$:PRINTINT(FRE(8)/M)"LINES FREE":STOP:GO
   TO6                                                   :rem 68
85 K$=CHR$(147):PRINTK$;R$"MARGIN"O$;:C$=CHR$(13):INPUTM:E=(8
   0-M)/2:GOTO6                                          :rem 66
90 L$="":IF(FRE(1)/M<2)THENPRINT"NO ROOM":GOTO6          :rem 64
91 PRINTCHR$(166);:GOSUB2:IFA$=C$THENPRINTX$;" ":RETURN
                                                        :rem 176
92 IFA$="@"THENA$="{5 SPACES}"                           :rem 53
93 PRINTX$;A$;:IFA$=D$THENL$=LEFT$(L$,LEN(L$)-1):GOTO91
                                                         :rem 22
94 IFLEN(L$)=M-5THENPOKEP,238:FORI=15TO1STEP-1:POKEP+3,I:NEXT
                                                        :rem 127
95 POKEP,0:L$=L$+A$:GOTO91                               :rem 13
```

Louis
Mendelsohn

VIC Marquee

The "VIC Marquee" will run on an unexpanded VIC and provides a moving billboard message across the top of the screen. The beauty of this marquee utility is that the moving display is generated as a wedge during the screen interrupt processing. This allows the main user program to be performing one task while the marquee is generated essentially as a background operation. The demonstration program combines simple animation with a marquee message.

The marquee subroutine is a machine language program which is POKEd into the cassette buffer using a BASIC loader.

Storing Strings

The VIC uses two techniques for storing string information. Strings which are defined as constants either by DATA statements or by declarations such as A$="MESSAGE" are saved in the program area. However, strings which are used in or formed by operations such as concatenation are saved in the string storage area. The start of this string area is stored in low byte/high byte form at locations 51 and 52, respectively. This method of string storage permits the user to create messages and generate marquees with a simple three-step procedure:

1. Form the message string by concatenation. It must terminate with a CHR$(0).
2. Provide the location of the message to the marquee program by transferring the data in locations 51 and 52 to locations 1011 and 1009, respectively.
3. Initiate the marquee by a SYS 1008.

Marquee Demo

The program "Marquee Demo" employs this technique to create a marquee display while an animated stick figure performs jumping jacks. The next paragraph discusses key elements of the program.

Line 5 clears the screen at program initialization. Line 11 sets the color memory of the first screen row to blue. In order for the marquee to be visible, it is necessary that the color be changed from white. Lines 21–24 set string data which will be used in composing the messages. As discussed previously, these strings are stored in the program area. Line 30 calls the subroutine which reads the machine language marquee program and POKEs it into memory. Line 41 creates the marquee message M$ by concatenation. The statement X=FRE (0) is included to make the VIC perform string garbage collection. This operation removes A$ and B$ from the string storage

area reducing memory usage. Line 42 calls the subroutine which POKEs the string location pointer into the marquee memory. Line 50 starts the marquee, while line 60 calls the subroutine which has an animated figure perform 50 jumping jacks. Line 61 performs a HOME so that when the jumping figure routine is called again, the animated figure will be properly positioned. Lines 71–90 duplicate the functions of lines 41–60.

This program scrolls your message across the top of the screen during the IRQ interrupt, the time during which your computer is processing keyboard input and other necessary tasks. The IRQ interrupt normally occurs every 1/60 second. Marquee adds a routine to be processed during this interrupt, independent of the BASIC program. In fact, if you press RUN/STOP, your message will continue to scroll across the screen while you modify the program. Adding a routine to the IRQ also causes the length of time needed to process the IRQ to be increased slightly.

This increase in time for IRQ processing can cause a slight increase in the execution time of a program. However, this routine will slow your BASIC program down by less than 2/10 second when using the unexpanded VIC.

To test this yourself, enter

99 PRINT "THIS IS THE END."

Then run the program while measuring the length of time from typing RUN until "THIS IS THE END." appears on the screen. Then delete lines 50 and 80, which call the Marquee routine, and time the program again. It may be necessary to make several test runs to obtain accurate timing.

The internal timer (TI$) can't be used, as it is updated during the IRQ interrupt. For more information on using interrupts in your programs, see *Programming the VIC*, by Raeto Collin West, from Compute! Books.

Marquee Demo

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

Some lines of this program require keywords to be abbreviated so that they will not exceed the four-screen-line limit. See Appendix B.

```
5 PRINT"{CLR}"                                          :rem 153
10 REM COLOR FOR MARQUEE                                :rem 187
11 FORI=0TO21:POKE38400+I,6:NEXTI                         :rem 5
20 REM MARQUEE MESSAGE STORAGE                          :rem 112
21 A$="THIS MESSAGE IS DISPLAYED WITHOUT REALLY SLOWING DOWN
   {SPACE}THE JUMPING MAN VERY MUCH."                   :rem 145
22 B$="{2 SPACES}THIS TECHNIQUE CAN BE USED IN GAMES OR VARIO
   US READING EXERCISES."                                :rem 84
23 C$="ANY SORT OF MESSAGE CAN BE PRINTED AS LONG AS THE MESS
   AGE IS LESS THAN 255"                                 :rem 94
24 D$=" CHARACTERS, INCLUDING THE CHR$(0) AT THE END OF THE M
   ESSAGE."                                              :rem 81
30 GOSUB800:REM MARQUEE LOADER                           :rem 96
```

```
40 REM SET-UP FIRST MARQUEE MESSAGE                    :rem 163
41 M$=A$+B$+CHR$(0):X=FRE(0)                           :rem 227
42 GOSUB2000                                           :rem 168
50 SYS1008:REM STARTS MARQUEE                           :rem 60
60 GOSUB200:REM JUMPING MAN                            :rem 140
61 PRINT"{HOME}"                                        :rem 75
70 REM SET-UP SECOND MARQUEE MESSAGE                   :rem 218
71 M$=C$+D$+CHR$(0):X=FRE(0)                           :rem 234
72 GOSUB2000                                           :rem 171
80 SYS1008:REM STARTS MARQUEE                           :rem 63
90 GOSUB200:REM JUMPING MAN                            :rem 143
100 END                                                :rem 104
200 REM THIS PROGRAM IS A JUMPING MAN                  :rem 153
205 PRINT:PRINT:PRINT"{RED}"                            :rem 18
210 FORM=1TO50                                          :rem 61
215 PRINT"{9 SPACES}MWN{2 SPACES}"                     :rem 219
220 PRINT"{10 SPACES}]{+}{3 SPACES}"                    :rem 11
225 PRINT"{9 SPACES}N M{2 SPACES}"                       :rem 5
230 FORT=1TO300:NEXTT                                   :rem 65
235 PRINT"{4 UP}"                                      :rem 175
240 PRINT"{10 SPACES}W{3 SPACES}"                       :rem 62
245 PRINT"{9 SPACES}N{+}M{2 SPACES}"                   :rem 173
250 PRINT"{10 SPACES}]{2 G}{2 SPACES}"                 :rem 178
255 FORT=1TO300:NEXTT                                   :rem 72
260 PRINT"{4 UP}"                                      :rem 173
265 NEXTM                                               :rem 41
270 PRINT"{BLU}{4 DOWN}"                               :rem 205
275 RETURN                                             :rem 126
800 REM MARQUEE BASIC LOADER                           :rem 165
801 FORAD=864TO1015:READ D:POKEAD,D:NEXTAD              :rem 68
864 DATA22,0,15,191,234,7                              :rem 253
870 DATA15,22,160,1,185,0                              :rem 245
876 DATA30,153,255,29,200,204                          :rem 198
882 DATA96,3,208,244,32,161                            :rem 107
888 DATA3,205,97,3,240,15                                :rem 7
894 DATA192,255,240,11,200,140                         :rem 242
900 DATA101,3,172,96,3,153                              :rem 41
906 DATA255,29,96,172,96,3                              :rem 71
912 DATA169,32,153,255,29,238                          :rem 212
918 DATA103,3,173,103,3,205                             :rem 88
924 DATA96,3,176,48,96,172                              :rem 74
930 DATA101,3,177,0,41,191                              :rem 38
936 DATA96,141,0,0,142,1                               :rem 197
942 DATA0,169,0,141,103,3                              :rem 242
948 DATA141,101,3,173,20,3                              :rem 38
954 DATA141,99,3,173,21,3                                :rem 4
960 DATA141,100,3,120,169,223                          :rem 185
966 DATA141,20,3,169,3,141                              :rem 47
972 DATA21,3,88,96,120,173                              :rem 59
978 DATA99,3,141,20,3,173                                :rem 9
```

```
984 DATA100,3,141,21,3,88                    :rem 251
990 DATA96,206,102,3,16,9                     :rem 5
996 DATA32,104,3,173,98,3                    :rem 10
1002 DATA141,102,3,108,99,3                  :rem 83
1008 DATA162,3,169,62,32,169                 :rem 152
1014 DATA3,96                                :rem 174
1016 RETURN                                  :rem 168
2000 REM MARQUEE MESSAGE LOCATION GENERATOR  :rem 187
2030 POKE1009,PEEK(52)                       :rem 199
2040 POKE1011,PEEK(51)                       :rem 192
2060 RETURN                                  :rem 168
```

Doug Smoak

Message Board

With "Message Board" and an unexpanded VIC, you'll be able to create messages of up to 212 characters that will scroll across the top of the screen. Since Message Board is driven by the interrupt routine, it can run the message independently of any other program or programming that may be going on.

When you run this program, you will be asked to give a stop code that will signal the end of your message entry and start the display. You should then *carefully* enter your message and the stop code. You can delete and backspace with the INST/DEL key while entering the message. If you have done everything right up to this point, the screen will clear, a READY prompt will appear, a blinking cursor will show on the screen, and your message should be "marching" across the top of the screen. If you have all of these things including asterisks (*) before and after your message, then you have done it all correctly.

If, instead, you have a VIC that is acting peculiarly, you probably have an error in the DATA statements that causes a problem in the machine language routine. Of course, you should always save a program that you have just typed in before you run it. Then you can recover from a crashed computer by powering down and then reloading to hunt for errors. The most likely reason for a crash or malfunction in the program would be incorrect DATA statements. Check them closely.

On the other hand, if the program has worked, you can now either type NEW and go on with other things or rerun and enter a new message. Either way, the old message will continue to move across the screen. You can actually watch it change as you enter the new message. But it will not be complete until you enter the stop code.

Hitting RUN/STOP–RESTORE will stop the message, but won't get rid of the machine language routine. It is protected along with your message in the top 335 bytes of RAM. To restart the message, just type SYS 7345 and press RETURN.

Programs which cause the screen to scroll may sometimes cause strange things to happen to the message. Furthermore, other programs that use the interrupt routine will cause problems. The message will also pause after a LIST or a STOP during another program's execution, but it has always started back after any of these pauses.

One more comment about problems that you may have entering the program: Be *very* sure to get lines 4, 12, and 13 right since the success of entering your message depends upon these lines.

Message Board

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
2 POKE56,28:POKE55,176:REM ** MEMORY POKES!! **          :rem 98
3 FORH=7345TO7419:READA:POKEH,A:NEXT                      :rem 24
4 I=7424:D=128:DEFFNA(K)=(KANDD)/2OR(KAND63)-(K>DANDK<193ORK<
  32ORK=95)*D                                            :rem 249
5 DN$="{HOME}{21 DOWN}":M$="{CLR}{RVS}{RED} ** MESSAGE
  {2 SPACES}BOARD ** {OFF}{BLU}"                         :rem 151
6 PRINTM$:GOSUB18                                         :rem 87
7 PRINTM$"{DOWN}ENTER A CHARACTER TO{2 SPACES}{DOWN}BE USED A
  S AN END OF{2 SPACES}{DOWN}MESSAGE CODE. NOT * !"      :rem 108
8 INPUTST$:IFST$="*"THEN7                                 :rem 56
9 FORJK=ITOI+22:POKEJK,170:NEXT:I=I+22                   :rem 137
10 PRINTM$"{DOWN}STOP CODE IS{SHIFT-SPACE}"ST$:PRINT"{DOWN}EN
   TER MESSAGE"                                          :rem 187
11 GETN$:ON-(N$="")GOTO11:IFN$=ST$THEN15                 :rem 111
12 IFN$="{DEL}"THENPRINTN$"{RVS} {OFF} {2 LEFT}";:M=M-1:GOTO1
   1                                                     :rem 200
13 PRINTN$"{RVS} {OFF}{LEFT}";:POKEI+M,FNA(ASC(N$))OR128:M=M+
   1:IFM=212THEN17                                        :rem 97
14 GOTO11                                                  :rem 0
15 FORJK=MTOM+22:POKEI+JK,170:NEXT                       :rem 154
16 POKE0,M+22:SYS7345:PRINT"{CLR}{RED}{RVS}{22 SPACES}{OFF}
   {BLU}":END                                            :rem 104
17 PRINT"MESSAGE TOO LONG!":SYS7345:END                  :rem 158
18 PRINTDN$"{4 SPACES}PRESS{2 SPACES}ANY KEY":GETA$:ON-(A$=""
   )GOTO18:RETURN                                        :rem 196
19 DATA120,169,3,160,22,153,255,149,136,192,0,208,248,234
                                                          :rem 43
20 DATA169,213,141,20,3,169,28,141,21,3,169,0,133,255,133,1,1
   69,29,133,2,88,96,230,255                             :rem 182
21 DATA165,255,201,10,208,28,160,0,177,1,153,0,30,200,192,22,
   208,246,230,1,165,1,197                                :rem 52
22 DATA0,208,4,169,0,133,1,169,0,133,255,76,191,234      :rem 244
```

Chapter Five

Sports Games

Tim
Adcock

Strike Three

"Strike Three' is, as you have probably guessed, a baseball game for the unexpanded VIC-20. This program will also run with a 3K expander. However, it will not run with any other memory expansions unless some modifications are made to all variables which reference a screen address. There are many of these variables, but they should not be hard to find. If you do not want to make these changes, just remove your expander (8K or 16K).

If you have an unexpanded VIC, type in the following line *instead* of lines 82–105:

82 GOTO 118

Also, you can delete line 57 if you still do not have enough memory on the standard VIC. If you do have a 3K expander, just type in the entire listing as shown.

Playing Strike Three

Once the program is run, a baseball diamond will appear on the screen. The usual baseball indicators, balls, strikes, and so forth, are shown at the bottom of the screen. SC1 stands for the first player's score, while SC2 is for player 2. You'll also notice another indicator displayed as RUNS. This category continuously shows the number of runs being scored in an inning. When a player finishes his or her half of the inning, RUNS will revert back to zero and start counting again for the other half of the inning.

Players choose who starts at bat. The player who pitches will use the period, comma, and slash keys:

KEY	Pitch
.	curve
,	change up
/	straight

When selecting a pitch, you may have to hit the key twice. The execution of the game may sometimes run a little slower than your selection.

After the pitch, the player at bat will use the space bar to swing the bat. If the batter misses the ball or lets it go by the plate with no swing, it is counted a strike. If the batter lets the ball go by on a curve pitch, it is counted a ball. Naturally, when any of these conditions exist, the indicators at the bottom will be updated. If the player hits the ball, you will see it go across the field to a somewhat random location.

As the ball is hit, the player who is pitching takes over. He or she will use the period key to move the fielder (the ball characters) right and the comma key to move the fielder left.

Because of the interaction between pitching and hitting, both players should hit their keys with a sort of tap. In other words, *don't hold a key down* as this could interfere with the other player. There are two exceptions to this rule: when a player is moving fielders and when a player is throwing the ball from the outfield. (The second exception applies only to the 3K expanded version of the game.)

As you are moving fielders, try to intercept the ball that has been hit. If you do intercept it, your fielder will turn red and you will have one out on the opposing player. If the ball is not caught, then it will turn red, either on the field or on the screen border, and the opposing player will advance to first base. Any other players already on base will also advance (this routine is also executed if a batter walks (takes four balls).

Expanded Version

Since we have a little extra memory in an expanded VIC, I've included a few extra features. As mentioned above, the listing as shown is the expanded version. Anyone with a standard VIC should delete lines 82–105 and insert a new line 82. The instructions for the "3K added" game are the same as for the other version except that the routine from 82 to 105 adds an extra feature to the game.

The period and comma keys are used to select a pitch, move fielders, *and* throw the ball. In the other version, you could *not* throw the ball. Here's how it works: When the ball is hit, the pitcher tries to move the fielders to intercept it. If the pitcher succeeds, the fielder turns red and an out is registered.

Now there is a slight time delay (this delay can be changed by changing the value of TI$). If the hitter does not have any players on base, just wait out the delay, and the routine between 82 and 105 is not used. If the hitter does have players on base, then the player at bat can still wait out the delay *or* can try to advance the players one base. This is done by pressing the space bar (don't hold it down). After this is done, the ball will leave the fielder who has caught it and will begin to move across the field. Again, this movement is controlled by the pitcher with the period and comma keys. As the ball moves, the pitcher should try to guide it to second, third, or home base. If it hits one of these three bases, the word *OUT* appears on the screen, and the player who would have made it to that base disappears.

I should also mention that this routine executes only if the fielder that caught the ball is an outfielder, not an infielder. Naturally, in real baseball, no one tries to advance on an infield fly.

Strike Three

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
0 CL=30720:S3=36876:S4=36877:S5=36878                    :rem 31
1 J1=1:J2=-1:J3=23:J4=21:O1=7772:O2=7734:O3=7784:OT=0:RN=0:GO
  TO39                                                  :rem 250
2 SS=0:BA=0:POKES3,210                                   :rem 29
3 POKEBL,32:C=0:FORT=1TO7                                :rem 40
4 FORX=1TOR                                              :rem 214
5 BL=BL+D1                                               :rem 46
6 IFPEEK(197)=37THENO2=O2+J1:O1=O1-J4                    :rem 185
7 IFPEEK(197)=37THENI1=I1-J4                             :rem 143
8 IFPEEK(197)=29THENO2=O2-J1:O1=O1+J4                    :rem 188
9 IFPEEK(197)=29THENI1=I1+J4                             :rem 144
10 IFPEEK(BL)=93THENC=2:T=7:X=R:GOTO15                   :rem 122
11 POKEBL,124:POKEO1,81:POKEO2,81:POKEO3,81:POKEI1,81:POKEI2,
   81                                                    :rem 80
12 IFBL=O2ORBL=O1THENOT=OT+1:T=7:X=R:C=1:GOTO15          :rem 31
13 IFBL=I1THENOT=OT+1:T=7:X=R:C=3:GOTO15                 :rem 47
14 POKEO2,32:POKEO1,32:POKEBL,32:POKEI1,32:POKES3,0      :rem 103
15 NEXT:IFT=7THEN18                                      :rem 206
16 BL=BL+D2:POKEBL,124:POKEBL,32:IFO1>7856THENO1=7878:rem 206
17 IFBL=O1THENOT=OT+1:T=7:C=1                            :rem 61
18 NEXT:POKES3,210:FORT=1TO30:NEXT:POKES3,0:IFC=0THEN106
                                                         :rem 66
19 ONCGOTO81,106,117                                     :rem 116
20 SS=0:BA=0:POKES3,210                                  :rem 77
21 POKEBL,32:C=0:FORT=1TO7                               :rem 88
22 FORX=1TOR                                             :rem 6
23 BL=BL+D1                                              :rem 94
24 IFPEEK(197)=37THENO2=O2+J1:O3=O3+J3                   :rem 234
25 IFPEEK(197)=37THENI2=I2+23                            :rem 166
26 IFPEEK(197)=29THENO2=O2-J1:O3=O3-J3                   :rem 241
27 IFPEEK(197)=29THENI2=I2-23                            :rem 171
28 IFO3>7877THENO3=7899                                  :rem 101
29 POKES3,0:IFPEEK(BL)=93THENC=2:T=7:X=R:GOTO34          :rem 208
30 POKEBL,124:POKEO1,81:POKEO2,81:POKEO3,81:POKEI1,81:POKEI2,
   81                                                    :rem 81
31 IFBL=O2ORBL=O3THENOT=OT+1:T=7:X=R:C=1:GOTO34          :rem 35
32 IFBL=I2THENOT=OT+1:T=7:X=R:C=3:GOTO34                 :rem 50
33 POKEO2,32:POKEO3,32:POKEBL,32:POKEI2,32               :rem 32
34 NEXT:IFT=7THEN37                                      :rem 208
35 BL=BL+D2:POKEBL,124:IFO3>7899THENO3=7921              :rem 71
36 IFBL=O3ORBL=O2THENOT=OT+1:T=7:C=1:GOTO37              :rem 10
37 POKEBL,32:NEXT:POKES3,210:FORT=1TO30:NEXT:POKES3,0:IFC=0TH
   EN106                                                 :rem 203
38 ONCGOTO81,106,117                                     :rem 117
39 PRINT"{CLR}{BLK}":POKE36879,93:POKES5,15              :rem 245
40 R=INT((RND(1)*5)+1):O1=7794:O2=7735:O3=7808:IFOT>=3THEN122
                                                         :rem 239
```

```
41 POKE8109,32:GOSUB119                                    :rem 76
42 B1=8005:B2=7867:B3=7993:HP=8131:I1=7907:I2=7915:PT=7998
                                                          :rem 218
43 FORJ=7888TO7972STEP21:POKEJ,78:NEXT                    :rem 150
44 FORJ=7890TO7982STEP23:POKEJ,77:NEXT                    :rem 146
45 FORJ=7921TO8110STEP21:POKEJ,78:NEXT                    :rem 125
46 FORJ=7878TO8108STEP23:POKEJ,77:NEXT                    :rem 145
47 POKEB1,160:POKEB2,160:POKEB3,160:POKEHP,160            :rem 213
48 POKEB1+CL,6:POKEB2+CL,6:POKEB3+CL,6:POKEHP+CL,6         :rem 58
49 POKEO1,81:POKEO2,81:POKEO3,81:POKEI1,81:POKEI2,81      :rem 161
50 POKE8130,103:POKE7999,108:FORT=1TO500:NEXT             :rem 196
51 FORT=7680TO7723:POKET,93:NEXT                          :rem 247
52 FORT=7701TO7899STEP22:POKET,93:NEXT                    :rem 160
53 FORT=7680TO7856STEP22:POKET,93:NEXT                    :rem 160
54 BL=7999+22:FORT=1TO10:GETB$:NEXT                       :rem 133
55 GETA$:IFA$=""THEN55                                    :rem 247
56 B1=124:B2=108:D=22:IFA$="."THENB1=126:B2=108:D=23       :rem 78
57 IFA$<>"/"ANDA$<>"."ANDA$<>","THEN55                     :rem 44
58 FORJ=1TO3:BB=124:POKEBL,BB:FORG=1TO5 :NEXT:POKEBL,32:BB=10
   8:POKEBL,BB                                             :rem 5
59 FORG=1TO5 :NEXT:POKEBL,32:BL=BL+22:POKES3,210-2*J:NEXTJ
                                                          :rem 73
60 IFPEEK(197)=32 THEN67                                   :rem 74
61 IFA$=","THENFORT=1TO250:NEXT                           :rem 147
62 BL=BL+D:POKEBL,B1:BB=1:FORG=1TO5 :NEXT:POKES3,202-Q:IFPEEK
   (197)=32THEN67                                         :rem 100
63 POKEBL,32:POKEBL,B2:FORG=1TO5 :NEXT:BB=2:Q=Q+2          :rem 24
64 IFPEEK(197)=32 THEN67                                   :rem 78
65 POKEBL,32:IFBL>8185THEN77                              :rem 193
66 GOTO62                                                  :rem 13
67 POKE8131,77:POKES3,0:Q=0:FORJ=1TO5:NEXT                :rem 112
68 POKE8131,99:FORJ=1TO10:NEXT:POKE8131,32                 :rem 37
69 POKE8109,78:FORJ=1TO10:NEXT                             :rem 97
70 POKE8109,101                                            :rem 38
71 IFBL=8131ANDBB=1THEND1=-21:D2=-22:GOTO20                :rem 24
72 IFBL=8109ANDBB=2THEND1=-22:D2=-21:GOTO20                :rem 31
73 IFBL=8087THEND1=-22:D2=-23:GOTO2                        :rem 49
74 IFBL=8109ANDBB=1THEND1=-23:D2=-22:GOTO2                :rem 242
75 FORT=1TO2000:NEXT:POKEBL,32:SS=SS+1:IFSS=3THENSS=0:BA=0:OT
   =OT+1                                                  :rem 254
76 GOTO40                                                  :rem 10
77 POKES3,0:Q=0:IFD=22THENSS=SS+1:IFSS=3THENSS=0:BA=0:OT=OT+1
   :GOTO40                                                :rem 116
78 IFD=22THEN40                                           :rem 118
79 BA=BA+1:IFBA=4THENBA=0:SS=0:GOTO106                     :rem 66
80 GOTO40                                                   :rem 5
81 POKEBL+CL,2:POKEBL,81 :TI$="000000":IFOT=>3THEN40       :rem 19
82 IFPEEK(197)=32THEN88                                    :rem 81
83 IFTI$>"000005"THEN104                                   :rem 38
84 IFBL=O1THENX=23:Y=-1:Z=-22                             :rem 255
85 IFBL=O2THENX=22:Y=-1:Z=1                               :rem 160
```

```
86 IFBL=O3THENX=21:Y=-22:Z=1                                  :rem 212
87 GOTO82                                                      :rem 18
88 D1=32:D2=32:D3=32:C1=7983:C2=7866:C3=8015                  :rem 116
89 IFPEEK(8006)=87THEND1=87                                    :rem 64
90 IFPEEK(7845)=87THEND2=87                                    :rem 67
91 IFPEEK(7992)=87THEND3=87                                    :rem 72
92 POKE8006,32:POKE7845,32:POKE7992,32                        :rem 160
93 FORE=1TO6:POKEC1,D1:POKEC2,D2:POKEC3,D3:POKES4,240 :rem 17
94 FORF=1TO3:BL=BL+X:POKEBL,124:FORG=1TO20:NEXTG:POKEBL,32:PO
   KES4,0                                                      :rem 29
95 IFPEEK(197)=29THENBL=BL+Y                                  :rem 200
96 IFPEEK(197)=37THENBL=BL+Z                                  :rem 201
97 IFBL=7867THENOT=OT+1:PRINT"{HOME}{6 DOWN}{10 RIGHT}{RVS}OU
   T{OFF}{UP}":X=0:Y=0:Z=0:D1=32                              :rem 216
98 IFBL=7993THENOT=OT+1:PRINT"{HOME}{6 DOWN}{10 RIGHT}{RVS}OU
   T{OFF}{UP}":X=0:Y=0:Z=0:D2=32                              :rem 218
99 IFBL=8131THENOT=OT+1:PRINT"{HOME}{6 DOWN}{10 RIGHT}{RVS}OU
   T{OFF}{UP}":X=0:Y=0:Z=0:D3=32                              :rem 205
100 NEXTF                                                      :rem 22
101 POKEC1,32:POKEC2,32:POKEC3,32:C1=C1-23:C2=C2+21:C3=C3+23:
    NEXTE                                                      :rem 54
102 POKEC1,D1:POKEC2,D2:POKEC3,D3:IFD3=87THENRN=RN+1 :rem 205
103 FORG=1TO300:NEXT:PRINT"{HOME}{6 DOWN}{8 RIGHT}{WHT}
    {10 SPACES}{BLK}"                                         :rem 224
104 FL=0:POKEO1,32:POKEO2,32:POKEO3,32:POKEI1,32:POKEI2,32
                                                              :rem 238
105 POKEO1+CL,1:POKEO2+CL,1:POKEO3+CL,1:GOTO40                 :rem 54
106 C1=8006:C2=7845:C3=7992:D1=32:D2=32:D3=32:HH=8153:POKES5,
    0:POKES4,235                                              :rem 115
107 IFPEEK(C1)=87THEND1=87                                     :rem 13
108 IFPEEK(C2)=87THEND2=87                                     :rem 16
109 IFPEEK(C3)=87THEND3=87                                     :rem 19
110 POKEBL,124:POKEBL+CL,2:FORT=1TO7:POKEHH,87:POKEC1,D1:POKE
    C2,D2:POKEC3,D3                                           :rem 242
111 FORG=1TO60:NEXT:POKEHH,32:POKEC1,32:POKEC2,32:POKEC3,32:P
    OKES5,T*2                                                  :rem 85
112 HH=HH-21:C1=C1-23:C2=C2+21:C3=C3+23:NEXTT:POKEBL,32:POKEB
    L+CL,1:POKES5,15                                           :rem 58
113 IFD3=87THENRN=RN+1                                         :rem 79
114 POKEHH,87:POKEC1,D1:POKEC2,D2                             :rem 238
115 FORG=7TO1STEP-1:POKES5,G*2:FORJ=1TO300:NEXT:NEXTG:POKES4,
    0:POKES5,15                                               :rem 126
116 GOTO40                                                     :rem 53
117 POKEBL,81:POKEBL+CL,2:FORT=1TO2000:NEXT:POKEBL+CL,1:POKEO
    1,32:POKEO2,32                                            :rem 190
118 POKEO3,32:POKEI1,32:POKEI2,32:GOTO40                      :rem 156
119 PRINT"{HOME}{20 DOWN}{RVS}RUNS ";RN;TAB(15)"{RVS}OUTS"OT;
    "{LEFT}"                                                   :rem 49
120 PRINT"{RVS}BALLS {LEFT}"BA;TAB(15)"{RVS}STRK"SS;"{LEFT}"
                                                               :rem 41
```

```
121 PRINTTAB(4)"{RVS}SC1"S1;TAB(13)"{RVS}SC2"S2;"{2 UP}":RETU
    RN                                                  :rem 67
122 IFII=ØTHENS1=S1+RN:RN=Ø:II=1:PRINT"{CLR}":OT=Ø:GOTO41
                                                       :rem 163
123 IFII=1THENS2=S2+RN:RN=Ø:II=Ø:PRINT"{CLR}":OT=Ø:GOTO41
                                                       :rem 166
```

Kerry Griffin

Speed Demon

The object of "Speed Demon," a game for the unexpanded VIC, is to drive your car around the track and try to complete as many laps as possible in two minutes.

If you complete more than ten laps, you'll get an extra minute of bonus time. When your time is up, you will be given a rating based on your performance. Work hard enough and maybe you will get a PRO rating.

In Speed Demon the A key is used to turn your car counterclockwise, and the D key to turn clockwise. Press f1 for high speed and f3 for low speed.

Saving the Programs

Speed Demon is divided into two programs. The first contains instructions and DATA statements for creating graphics. If you use tape, enter Program 1 with these changes: Delete lines 10010 and 10020 and substitute the following line for line 10000:

10000 POKE36879,15:PRINT"{HOME}{YEL}PLEASE WAIT":POKE198,2:POKE631,131:END

Save Program 1 to tape. Then key in Program 2 and save it immediately after Program 1. When you run the first program, it will automatically load and run Program 2.

If you use disk, be sure to save Program 2 with the filename SPEED.1, or adjust line 10010 of Program 1 to match your filename.

Program 1. Speed Demon

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}"                                          :rem 197
20 PRINT"{5 SPACES}{PUR}SPEED DEMON"                     :rem 179
30 PRINT"{5 SPACES}{11 T}                                 :rem 19
40 PRINT" {BLK}DRIVE YOUR CAR AROUND THE TRACK AND TRY TO
   {2 SPACES}COMPLETE AS MANY LAPS AS ";                  :rem 25
50 PRINT"POSSIBLE IN 2{6 SPACES}MINUTES."               :rem 179
60 PRINT"{DOWN} COMPLETE MORE THAN 10 LAPS AND GET 1 MINUTE B
   ONUS TIME."                                           :rem 42
65 PRINT"{DOWN} PRESS 'A' TO ROTATE{3 SPACES}YOUR CAR COUNTER
   {6 SPACES}CLOCKWISE AND 'D' TO{2 SPACES}TURN";        :rem 80
66 PRINT" CLOCKWISE."                                    :rem 15
67 PRINT" PRESS 'F1' TO GO FAST AND 'F3' TO GO SLOW." :rem 61
70 PRINT"{DOWN}PRESS ANY KEY TO GO ON"                  :rem 125
80 GETA$:IFA$=""THEN80                                  :rem 243
90 PRINT"{CLR}{BLU}{RVS} WAIT A MINUTE"                  :rem 70
997 POKE52,28:POKE56,28                                  :rem 63
998 FORI=7168TO7679:POKEI,PEEK(I+25600):NEXT            :rem 171
```

```
999 POKE36869,255                                          :rem 178
1000 READN:IFNTHENFORN=NTON+15:READA:POKEN,A:NEXT:GOTO1000
                                                            :rem 27
9330 DATA 7432 ,0,56,84,124,56,124,124,56,56,124,124,56,124,8
     4,56,0                                                 :rem 189
9350 DATA 7448 ,0,54,95,127,95,54,0,0,0,108,250,254,250,108,0
     ,0                                                     :rem 229
9370 DATA 7464 ,48,88,184,254,127,31,30,12,12,26,29,127,254,2
     48,120,48                                              :rem 98
9390 DATA 7480 ,12,30,31,127,254,184,88,48,48,120,248,254,127
     ,29,26,12                                              :rem 98
9410 DATA 7496 ,230,230,152,152,230,230,128,128,8,28,28,62,62
     ,127,8,8                                               :rem 34
9515 DATA7512,254,253,251,247,239,223,191,127,127,191,223,239
     ,247,251,253,254                                       :rem 37
9520 DATA 7528,126,189,219,231,231,219,189,126,255,255,255,25
     5,255,255,255,255                                      :rem 58
9525 DATA7640,127,127,127,127,127,127,127,127,254,254,254,254
     ,254,254,254,254                                       :rem 40
9535 DATA7656,0,255,255,255,255,255,255,255,255,255,255,255,2
     55,255,255,0                                          :rem 112
9540 DATA7392,68,8,34,136,5,64,17,68,0,0,0,0,0,0,0,0  :rem 29
9999 DATA 0                                                 :rem 46
10000 POKE36879,15:PRINT"{CLR}{YEL} PLEASE WAIT":POKE 198,2:P
      OKE631,13:POKE 632,13                                :rem 126
10010 PRINT "{HOME}{BLK}":PRINT "{2 DOWN}LOAD";CHR$(34);"SPEE
      D.1";CHR$(34);",8":PRINT "{4 DOWN}RUN"               :rem 17
10020 PRINT "{HOME}":END                                  :rem 232
```

Program 2. Speed.1

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 POKE 36879,27:PRINT"{CLR}"                              :rem 164
5 S=7739:SS=7761:C=30720:PA=37137:PX=36872:POKE36878,7:S1=368
  77:D=1:V=S                                              :rem 111
7 POKE36877,0                                             :rem 209
8 T=75:T$="1"                                             :rem 153
10 PRINT"{CLR}{PUR}{5 SPACES}SPEED DEMON"                  :rem 69
20 PRINT"{RVS}{5 SPACES}[11 T]"                            :rem 70
30 PRINT"{5 DOWN} HIT ANY KEY AND THEN{2 SPACES}WHEN THE CHEC
   KERED{4 SPACES}FLAG APPEARS{RVS}...{OFF}GO"            :rem 188
40 GETA$:IFA$=""THEN40                                    :rem 235
50 GOSUB1000                                              :rem 166
55 PRINT"{HOME}{2 SPACES}{BLU}LAPS:"L"{2 SPACES}TIME:"T
                                                           :rem 36
60 POKES,35:POKES+C,4                                       :rem 6
70 POKE7736,41:POKE7736+C,0                                 :rem 8
80 FORO=1TO500:NEXTO                                       :rem 12
85 J=200                                                  :rem 134
90 POKE7736,32                                              :rem 0
```

```
100 TI$="000000"                                          :rem 243
110 M$=MID$(TI$,4,1)                                       :rem 13
120 S$=MID$(TI$,5,1)                                       :rem 21
130 O$=MID$(TI$,6,1)                                       :rem 19
137 IFM$>T$THEN10000                                      :rem 113
140 PRINT"{HOME}{7 RIGHT}"L"{7 RIGHT}"M$":"S$O$           :rem 120
143 POKE 36878,7                                           :rem 58
145 POKE36877,J                                            :rem 78
160 IFPEEK(197)=18THEND=D-1                                :rem 53
180 IFPEEK(197)=17THEND=D+1                                :rem 52
185 IFPEEK(197)<>1ANDPEEK(197)<>4THEND=D                  :rem 123
190 IFD>8THEND=1                                          :rem 196
200 IFD<1THEND=8                                          :rem 186
210 IFD=1THENG=35:Z=-1                                     :rem 30
220 IFD=2THENG=39:Z=+21                                    :rem 84
230 IFD=3THENG=34:Z=+22                                    :rem 82
240 IFD=4THENG=40:Z=+23                                    :rem 82
250 IFD=5THENG=36:Z=+1                                     :rem 37
260 IFD=6THENG=38:Z=-21                                    :rem 93
270 IFD=7THENG=33:Z=-22                                    :rem 91
280 IFD=8THENG=37:Z=-23                                    :rem 98
290 IFPEEK(V+Z)<>32ANDPEEK(V+Z)<>160THEN5000             :rem 131
300 POKEV+Z,G:POKEV+Z+C,4:POKEV,32:V=V+Z                   :rem 26
310 IFV=SORV=SSTHENL=L+1                                   :rem 67
330 IFPEEK(197)=39THENT=0:J=225                            :rem 47
340 IFPEEK(197)=47THENT=75:J=200                          :rem 100
350 FORE=0TOT:NEXTE                                       :rem 230
360 GOTO110                                               :rem 100
1000 PRINT"{CLR}"                                          :rem 37
1010 PRINT"{OFF}{GRN}** *{2 SPACES}**{BLK}{RVS}{11 @}{OFF}
     {GRN}**"                                             :rem 199
1020 PRINT"{OFF}{GRN} *{2 SPACES}***{BLK}{RVS}N{11 SPACES}M
     {OFF}{GRN}*"                                           :rem 3
1025 PRINT"{OFF}{GRN}* ** *{BLK}{RVS}N{13 SPACES}{G}{OFF}
     {GRN}"                                               :rem 182
1030 PRINT"{OFF}{GRN}*{2 SPACES}**{BLK}{RVS}N{2 SPACES}N{7 T}
     M{3 SPACES}{G}{OFF}{GRN}"                            :rem 152
1040 PRINT"{OFF}{GRN} ** {BLK}{RVS}N{2 SPACES}N{OFF}{GRN}** *
     * * {RVS}{BLK}{M}{3 SPACES}{G}{OFF}{GRN}"            :rem 248
1050 PRINT"{OFF}{GRN}* *{BLK}{RVS}N{2 SPACES}N{OFF}{GRN}
     {BLK},>+{GRN}*{2 SPACES}**{RVS}{BLK}N{3 SPACES}{G}{OFF}
     {GRN}"                                                 :rem 15
1060 PRINT"* {BLK}{RVS}N{2 SPACES}N{OFF}{GRN}{2 SPACES}{BLK}+
     =,{GRN}*{RVS}{BLK}{3 @}N{4 SPACES}{G}{OFF}{GRN}":rem 205
1070 PRINT" {BLK}{RVS}N{2 SPACES}N{OFF}{GRN}* ** {RIGHT}{RVS}
     {BLK}N{7 SPACES}N{OFF}{GRN}"                           :rem 16
1080 PRINT"{BLK}{RVS}{M}{3 SPACES}{G}{OFF}{GRN}***{2 SPACES}
     {RVS}{BLK}{M}{7 SPACES}N{OFF}{GRN} *"                :rem 167
1090 PRINT"{BLK}{RVS}{M}{3 SPACES}{G}{OFF}{GRN}** * {RVS}
     {BLK}{M}{2 SPACES}N{4 T}{OFF}{GRN} * "                 :rem 52
```

```
2000 PRINT"{BLK}{RVS}[M]{3 SPACES}[G]{OFF}{GRN}* ** {RVS}
     {BLK}[M]{2 SPACES}[G]{OFF}{GRN}*** {BLK},+{GRN}*"
                                                    :rem 250
2010 PRINT"{BLK}{RVS}[M]{3 SPACES}[G]{OFF}{GRN} ** *{RVS}
     {BLK}[M]{2 SPACES}M{OFF}{GRN}**{2 SPACES}{BLK}<;{GRN} "
                                                    :rem 239
2020 PRINT"{BLK}{RVS}[M]{3 SPACES}[G]{OFF}{GRN}***{2 SPACES}*
     {RVS}{BLK}M{2 SPACES}M{OFF}{GRN}*{2 SPACES}{BLK}+,{GRN}*
     "                                              :rem 32
2030 PRINT"{BLK}{RVS}[M]{3 SPACES}[G]{OFF}{GRN} **{2 SPACES}*
     *{RVS}{BLK}M{2 SPACES}M{OFF}{GRN} **{2 SPACES}"  :rem 28
2040 PRINT"{BLK}{RVS}[M]{2 SPACES}N{OFF}{GRN}***{BLK},.+{GRN}
     {2 SPACES}*{RVS}{BLK}M{2 SPACES}M{OFF}{GRN}**{2 SPACES}"
                                                    :rem 121
2050 PRINT"{BLK}{RVS}[M]{2 SPACES}[G]{OFF}{GRN}*{2 SPACES}
     {BLK}.-.{GRN}{2 SPACES}* {RVS}{BLK}M{2 SPACES}M{OFF}
     {GRN}** "                                        :rem 1
2060 PRINT"{BLK}{RVS}[M]{2 SPACES}[G]{OFF}{GRN}*{2 SPACES}
     {BLK}+.,{GRN}{2 SPACES}* {RVS}{BLK}[M]{3 SPACES}[G]{OFF}
     {GRN}**"                                       :rem 176
2070 PRINT"{BLK}{RVS}[M]{2 SPACES}[G]{OFF}{GRN}**{3 SPACES}**
     *{2 SPACES}{RVS}{BLK}[M]{3 SPACES}[G]{OFF}{GRN}**"
                                                    :rem 252
2080 PRINT"{BLK}{RVS}[M]{2 SPACES}M[10 @]N{2 SPACES}N{OFF}
     {GRN}**"                                       :rem 185
2090 PRINT"{GRN}*{RVS}{BLK}M{14 SPACES}N{OFF}{GRN}{2 SPACES}*
     *"                                             :rem 37
3000 PRINT"{GRN} *{RVS}{BLK}M{12 SPACES}N{OFF}{GRN}***
     {2 SPACES}"                                    :rem 71
3010 PRINT"{GRN}* *{RVS}{BLK}[12 T]{OFF}{GRN}***{3 SPACES}
     {UP}"                                          :rem 12
3020 RETURN                                         :rem 165
5000 POKEV,G                                        :rem 189
5005 FORT=1TO100:NEXTT                              :rem 116
5007 POKEV,28:POKEV+C,2                             :rem 114
5010 POKE36877,220                                  :rem 196
5020 FORE=15TO0STEP-5                                :rem 7
5025 POKE36878,E                                    :rem 124
5030 FORA=1TO300:NEXTA                               :rem 78
5040 NEXTE                                           :rem 77
5045 POKEV,32:V=7738:POKEV,35:POKEV+C,4:D=1          :rem 86
5050 GOTO110                                        :rem 149
10000 IFL>10THENPRINT"{HOME}{DOWN}{RIGHT}{RVS}BONUS {HOME}
      {2 DOWN}{RIGHT}TIME!":T$="0":TI$="000000":GOTO110
                                                    :rem 235
10001 POKE36877,0:PRINT"{CLR}{PUR}{5 SPACES}TIME IS UP"
                                                     :rem 54
10002 IFL<6THENR$="GRANNY"                           :rem 53
10003 IFL>5ANDL<10THENR$="ROOKIE"                   :rem 237
10004 IFL>10ANDL<20THENR$="AMATEUR"                  :rem 97
10009 IFL>20THENR$="PRO"                            :rem 140
```

```
10010 PRINT"{4 DOWN} YOU COMPLETED "L"LAPS"                :rem 97
10015 PRINT"{2 DOWN}YOUR RATING IS {BLK}"R$               :rem 160
10020 PRINT"{3 DOWN}{PUR}{4 RIGHT} PLAY AGAIN?"           :rem 220
10025 S=7739:SS=7761:C=30720:POKE36878,7:S1=36877:D=1:V=S
                                                          :rem 252
10027 POKE198,0:L=0                                        :rem 26
10030 GETA$:IFA$=""THEN10030                               :rem 11
10040 IFA$="Y"THEN5                                        :rem 39
10050 PRINT"{CLR}":POKE36879,27:POKE36869,240:PRINT"{BLU}":EN
      D                                                   :rem 223
```

Jeff
Ranney

Lap Racer

Driving games can be exciting and challenging, and provide a pleasant change from the usual "blast the alien" games. Commodore has a few of these for the VIC-20, but as far as I know there is no good old-fashioned game where the driver races around a track for the best time.

Knowing this, I developed a program for the unexpanded VIC called "Lap Racer." The objective is to race around the track anywhere from one to nine times (chosen at the beginning of the game) and beat the clock.

The race car is controlled with the joystick. It is not limited to vertical and horizontal movements only, but can also move diagonally. The fire button is used to accelerate. You may circle the track either way you like, but you must hold that direction throughout the game.

Understanding the Program

Lines 90–110 initialize the variables. The variable A contains the location of the car on the screen (1–506). SQ is the beginning of the POKE locations on the screen (7680), and SW is the color location (38400).

Lines 500–530 contain the joystick subroutine. These lines check to see which way the joystick is pointed, then adjust A accordingly.

Line 550 POKEs in the race car. JS(X+4,Y+1) is the current direction that the car is pointing.

Lines 1300–1440 PRINT the race track onto the screen. By changing these lines, the user can put obstacles on the track or can draw a completely new one. A good way to create new race tracks is to draw them out on graph paper (22 × 23) first, then type in the PRINT statements.

Creating Programmable Characters

Lines 3000–5000 design the specialized characters used in this program. Line 3090 POKEs the codes for these characters into the new character memory. CH is the location of the new character memory (7168). Z*8 is the character that must be changed, and J is the byte in that character (each character contains eight bytes) that is being changed.

The first number in each DATA statement from line 4010 to line 4050 is the POKE code of the character you are changing. For example, the first number in line 4010 is 3. The POKE code for the letter *C* is 3, so line 4010 is changing the letter *C*. Now, if you ask the computer to print a *C* later in the program, it will print a race car. The POKE code for *D* is 4, for *E*, it is 5, and so forth.

The rest of the numbers in the DATA statements are the values of each of the eight bytes of the character you are changing. Each byte in that

character controls a horizontal line on the screen that is eight pixels long. These bytes contain eight bits, each of which is simply 1 or 0, on or off. If a bit is 1, then the corresponding pixel is turned on. Since the bits can be only 1 or 0, values in the bytes must be represented in binary. For example, the value 255 is represented in binary as eight 1's, or 11111111. If a byte in character memory contains a value of 255, then all the bits in that byte would be turned on, turning on all eight pixels, thus creating a solid horizontal line in that character.

The code for a race car pointing upward is contained in line 4010, starting with the number 90, which is the binary code for the top line of the race car, and ending with 60, which is the code for the bottom line. Lines 4010–4040 are the race cars (pointing in different directions), and line 4050 is the explosion character.

When line 3080 reaches the DATA statement in line 5000, it tells the computer:

POKE 36869,255

This changes the character memory pointer from a point in ROM (the part of the computer that the user can't access) to a point in RAM, namely, 7168.

Once you have changed the character memory pointer, however, the computer will look at the new character memory for *all* the character codes. Consequently, if the computer is asked to print something later in the program, it is going to print a bunch of unreadable garbage. Don't panic: There happens to be an easy solution to the problem. Simply print the *reverse* of whatever you want to print, and it will do what you wanted in the first place. To print the reverse of something, just hit CTRL and 9 inside quotation marks, and the rest of the characters will be reversed.

Lap Racer

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
40 POKE36878,3:A=200:PRINT"{CLR}"                          :rem 234
90 SQ=7680:SW=38400                                         :rem 63
100 DIMJS(5,2):POKE37139,0:DD=37154:PA=37137:PB=37152:rem 211
110 FORI=0TO2:FORJ=0TO5:READJS(J,I):NEXTJ,I                :rem 239
120 DATA-23,-22,-21,3,3,5,-1,0,1,6,3,5,21,22,23,6,4,4 :rem 14
160 GOSUB3000                                              :rem 218
180 SE=36877                                               :rem 125
340 PRINT"{RVS}HOW MANY LAPS?(1-9)"                        :rem 244
342 GETCV:IFCV=0THEN342                                    :rem 169
350 GOSUB1300                                              :rem 220
370 V=TI                                                   :rem 202
500 POKEDD,127:S3=-((PEEK(PB)AND128)=0):POKEDD,255         :rem 122
510 P=PEEK(PA):S1=-((PAND8)=0):S2=((PAND16)=0):S0=((PAND4)=0)
                                                           :rem 177
520 FR=-((PAND32)=0):X=S2+S3:Y=S0+S1                        :rem 55
525 POKESQ+A,32:POKESE,210+FR*27                           :rem 130
```

```
530 A=A+JS(X+1,Y+1):IFA<22THENA=A+22                    :rem 44
535 FORI=1TO27-FR*27:NEXT                               :rem 24
540 IFPEEK(SQ+A)<>32THEN600                             :rem 82
550 POKESQ+A,JS(X+4,Y+1):POKESW+A,2                    :rem 104
555 IFA>483THENA=A-22                                  :rem 203
569 GOTO500                                            :rem 114
600 POKESE,0                                           :rem 185
602 IFPEEK(SQ+A)=194THEN730                             :rem 81
603 POKESQ+A,10:POKESW+A,6                              :rem 70
620 POKESE,220                                          :rem 31
630 FORL=15TO0STEP-1:POKE36878,L                       :rem 205
635 FORI=1TO100:NEXT:NEXT                               :rem 98
638 POKESE,0:POKE36878,2:PRINT"{CLR}"                   :rem 57
640 GOSUB1300                                          :rem 222
642 PRINT"{HOME}{RVS}"SU                                :rem 58
645 GOTO500                                            :rem 109
730 SU=SU+1:PRINT"{HOME}{RVS}"SU                        :rem 91
735 TM=TI-V                                            :rem 157
740 IFSU=CVTHENPRINT"{CLR}{RVS}TIME ELAPSED{HOME}{2 DOWN}FOR"
    CV"LAPS="TM:POKE36869,240:GOTO745                  :rem 207
742 GOTO750                                            :rem 114
745 :PRINT"{DOWN}HIT ANY KEY"                          :rem 114
746 GETR$:IFR$=""THEN746                               :rem 135
747 RUN                                                :rem 151
750 POKESQ+A,194:POKESW+A,5:A=A+JS(X+1,Y+1):GOTO569     :rem 67
1300 POKE36879,124                                     :rem 199
1302 PRINT"{PUR}IIIIIIIIIIIIIIIIIIIII"                  :rem 48
1305 PRINT"I{6 SPACES}IIIIIIIIIIIII"                   :rem 225
1310 PRINT"I{6 SPACES}III{8 SPACES}III"                :rem 149
1315 PRINT"I{6 SPACES}II{9 SPACES}III"                  :rem 81
1320 PRINT"I{3 SPACES}I{2 SPACES}I{4 SPACES}III{4 SPACES}II"
                                                       :rem 223
1325 PRINT"I{3 SPACES}I{2 SPACES}I{3 SPACES}IIII{5 SPACES}I"
                                                       :rem 228
1330 PRINT"I{2 SPACES}II{2 SPACES}I{2 SPACES}IIIIII{4 SPACES}
     I"                                                :rem 187
1335 PRINT"I{2 SPACES}II{2 SPACES}I{2 SPACES}IIIIIII
     {3 SPACES}I"                                       :rem 9
1340 PRINT"I{2 SPACES}II{2 SPACES}{RVS}{GRN}B{OFF}{PUR}
     {2 SPACES}IIIIIII{3 SPACES}I"                     :rem 220
1345 PRINT"I{2 SPACES}II{2 SPACES}{RVS}{GRN}B{OFF}{PUR}
     {2 SPACES}IIIIIII{3 SPACES}I"                     :rem 225
1350 PRINT"I{2 SPACES}III {RVS}{GRN}B{OFF}{PUR} IIIIIIII
     {3 SPACES}I"                                      :rem 111
1355 PRINT"I{2 SPACES}IIIIIIIIII{6 SPACES}I"            :rem 84
1360 PRINT"I{2 SPACES}IIIIIIII{8 SPACES}II"            :rem 190
1365 PRINT"I{3 SPACES}IIIIII{8 SPACES}III"             :rem 122
1370 PRINT"II{3 SPACES}IIIII{3 SPACES}IIIIIIII"        :rem 227
1375 PRINT"III{3 SPACES}IIII{3 SPACES}IIIIIIII"        :rem 232
1380 PRINT"III{3 SPACES}IIIII{6 SPACES}IIII"             :rem 9
```

```
1385 PRINT"III{3 SPACES}IIIIII{6 SPACES}III"          :rem 14
1390 PRINT"III{15 SPACES}III"                         :rem 84
1395 PRINT"IIII{13 SPACES}IIII"                      :rem 235
1400 PRINT"IIIII{11 SPACES}IIIII"                    :rem 112
1430 PRINT"{RVS}{RED}QQQQQQQQQQQQQQQQQQQQ"           :rem 236
1440 RETURN                                          :rem 169
2999 REM DESIGN CUSTOM CHARACTERS                     :rem 54
3000 CH=7168:POKE51,240:POKE52,CH/256-1:POKE55,240:POKE56,CH/
     256-1                                           :rem 237
3020 FORI=0TO7:POKECH+256+I,0:NEXT                    :rem 65
3080 READZ:IFZ=-1THENPOKE36869,255:RETURN             :rem 79
3090 FORJ=0TO7:READB:POKECH+Z*8+J,B:NEXTJ             :rem 93
4000 GOTO3080                                        :rem 200
4010 DATA3,90,126,90,24,189,255,189,60               :rem 137
4020 DATA4,60,189,255,189,24,90,126,90               :rem 139
4030 DATA5,112,39,242,255,255,242,39,112             :rem 222
4040 DATA6,14,228,79,255,255,79,228,14               :rem 146
4045 DATA9,255,255,255,255,255,255,255,255             :rem 96
4050 DATA10,201,107,62,252,63,124,214,147              :rem 3
5000 DATA-1                                           :rem 61
```

Chapter Six

Logic and Luck

Richard F. Bohr

Slot Machine

"Slot Machine" is an interesting game for the unexpanded VIC. Writing the game, I gained a lot of experience with cursor and subroutine usage on the VIC. Additionally, I discovered some interesting sound effects and a method by which I was able to blink a prompt on the display.

Similar to a real slot machine, this version allows a combination of five ways to win. You select the number of ways to win by depressing the fire button on the joystick. Each depression of the fire button gives another way to win. After you make your winning selections, the joystick handle is "pulled" for the slot machine to run.

Selection of winning combinations is purely random with a weighting factor toward the least payoff. Lines 53–58 show how this weighting factor is accomplished and how the random numbers are manipulated. Winning payoffs are as follows:

Red Ball	15
White Diamond	50
Purple Club	100
Cyan Heart	150
Blue Spade	500
Green Diamond	2500
Black Bar	5000

To start the game, depress the fire button on the joystick and have some fun with a slot machine on your own VIC-20.

The Program

For those who are interested in the usage of the joystick fire button and handle, lines 30–49 will show the techniques I used. Also included in this logic are the necessary checks to insure that the pot doesn't decrease below zero and to limit the number of fire button depressions to five.

Graphic generation is accomplished in lines 1500–1630. Display of the graphics occurs in lines 70–106 along with the random number generators and weighting factor.

Slot Machine

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 DIMT$(7):T$(1)="{RED}Q{BLU}":T$(2)="{WHT}Z{BLU}":T$(3)="
  {PUR}X{BLU}":T$(4)="{CYN}S{BLU}":T$(5)="{BLU}A{BLU}"
                                                    :rem 203
2 T$(6)="{GRN}Z{BLU}":T$(7)="{BLK}{+}{BLU}"         :rem 217
3 W3=7:W2=1:W1=2:W6=7:W5=1:W4=2:W9=7:W8=1:W7=2      :rem 226
```

```
4 P=200:S=0:V=36878:N=36875:AC=1                                  :rem 40
5 POKE36879,126:POKE37139,0                                        :rem 6
6 PRINT"{CLR}":PRINTTAB(5)"SLOT MACHINE":PRINT:PRINT:PRINT
                                                                 :rem 142
7 FORI=1TO7                                                      :rem 175
8 PRINTTAB(8)"{WHT}{RVS}{7 SPACES}{OFF}{BLU}"                     :rem 49
9 NEXTI                                                          :rem 193
10 PRINT"{6 UP}"TAB(9)T$(1)TAB(11)T$(1)TAB(13)T$(1)              :rem 253
11 PRINT:PRINTTAB(9)T$(7)TAB(11)T$(7)TAB(13)T$(7)                 :rem 45
12 PRINT:PRINTTAB(9)T$(1)TAB(11)T$(1)TAB(13)T$(1)                 :rem 28
13 PRINT:PRINT:PRINT:PRINTTAB(7)"POT="P                          :rem 105
14 PRINT"{HOME}{20 DOWN}"TAB(2)"SELECT WAYS TO WIN"              :rem 208
15 GOTO30                                                          :rem 2
16 PRINT"{HOME}{3 DOWN}"TAB(6)"{RED}*{DOWN}M{8 DOWN}"TAB(15)"
   {RED}M"                                                       :rem 135
17 RETURN                                                          :rem 72
18 PRINT"{HOME}{6 DOWN}"TAB(6)"{RED}**"TAB(15)"*{BLU}":rem 12
19 RETURN                                                          :rem 74
20 PRINT"{HOME}{8 DOWN}"TAB(6)"{RED}**"TAB(15)"*{BLU}":rem 39
21 RETURN                                                          :rem 67
22 PRINT"{HOME}{10 DOWN}"TAB(6)"{RED}**"TAB(15)"*{BLU}"
                                                                   :rem 75
23 RETURN                                                          :rem 69
24 PRINT"{HOME}{13 DOWN}"TAB(6)"{RED}*{UP}N{8 UP}"TAB(15)"N
   {BLU}"                                                        :rem 181
25 RETURN                                                          :rem 71
26 S=S+1:POKE36878,15:POKE36875,195                              :rem 243
27 ONSGOSUB20,22,18,24,16                                        :rem 133
28 POKE36878,0:POKE36875,0                                       :rem 215
29 RETURN                                                          :rem 75
30 FB=((PEEK(37137)AND32)=0)                                     :rem 234
31 IFP=0ANDS=0THEN123                                               :rem 8
32 IFP=0THEN34                                                     :rem 71
33 IFFBTHENGOTO39                                                  :rem 81
34 HA=((PEEK(37137)AND8)=0)                                      :rem 194
35 IFS=0THEN30                                                     :rem 73
36 IFHATHENGOTO48                                                  :rem 85
37 AC=1                                                            :rem 92
38 GOTO30                                                           :rem 7
39 IF(AC=1ANDS<>5)THENGOSUB26:P=P-5                              :rem 163
40 IFS=5ORP=0THEN44                                              :rem 173
41 FR=FRE(0):PRINT"{HOME}{16 DOWN}"TAB(5)"PULL HANDLE":rem 80
42 PRINT"{HOME}{18 DOWN}"TAB(9)"OR"                              :rem 194
43 GOTO45                                                           :rem 9
44 PRINT"{HOME}{18 DOWN}"TAB(9)"{3 SPACES}{2 DOWN}{11 LEFT}
   {20 SPACES}"                                                     :rem 4
45 PRINT"{HOME}{14 DOWN}{BLU}"TAB(12)"{5 SPACES}{5 LEFT}"P
                                                                 :rem 138
46 AC=0                                                            :rem 91
47 GOTO30                                                           :rem 7
48 A=INT(RND(1)*7)+1                                               :rem 73
```

```
49 IFA<2THENA=2                                          :rem 138
50 PRINT"{HOME}{16 DOWN}"TAB(5)"{12 SPACES}"             :rem 250
51 PRINT:PRINTTAB(8)"{3 SPACES}"                          :rem 94
52 PRINT:PRINTTAB(2)"{19 SPACES}"                         :rem 89
53 D=7:FORI=1TO6:GOSUB60:NEXTI                           :rem 179
54 D=A:FORI=1TO3:GOSUB60:NEXTI                           :rem 187
55 D=7:FORI=1TO3:GOSUB69:NEXTI                           :rem 187
56 D=A:FORI=1TO3:GOSUB69:NEXTI                           :rem 198
57 D=7:FORI=1TO3:GOSUB78:NEXTI                           :rem 189
58 D=A:FORI=1TO3:GOSUB78:NEXTI                           :rem 200
59 GOTO89                                                 :rem 24
60 X=INT(RND(1)*D)+1:W3=W2:W2=W1:W1=X                    :rem 208
61 POKE36875,215:POKE36876,187:POKE36878,15              :rem 181
62 POKE36878,0                                             :rem 3
63 PRINT"{HOME}"TAB(9)"{6 DOWN}";                        :rem 147
64 K=W3:GOSUB88                                          :rem 166
65 PRINT"{HOME}"TAB(9)"{8 DOWN}";                        :rem 183
66 K=W2:GOSUB88                                          :rem 167
67 PRINT"{HOME}"TAB(9)"{10 DOWN}";                       :rem 219
68 K=W1:GOSUB88                                          :rem 168
69 Y=INT(RND(1)*D)+1                                     :rem 113
70 POKE36875,219:POKE36876,225:POKE36878,15:POKE36878,0
                                                         :rem 135
71 W6=W5:W5=W4:W4=Y                                      :rem 167
72 PRINT"{HOME}"TAB(11)"{6 DOWN}";                       :rem 188
73 K=W6:GOSUB88                                          :rem 169
74 PRINT"{HOME}"TAB(11)"{8 DOWN}";                       :rem 224
75 K=W5:GOSUB88                                          :rem 170
76 PRINT"{HOME}"TAB(11)"{10 DOWN}";                        :rem 4
77 K=W4:GOSUB88                                          :rem 171
78 Z=INT(RND(1)*D)+1                                     :rem 114
79 POKE36875,223:POKE36876,227:POKE36878,15:POKE36878,0
                                                         :rem 141
80 W9=W8:W8=W7:W7=Z                                      :rem 183
81 PRINT"{HOME}"TAB(13)"{6 DOWN}";                       :rem 190
82 K=W9:GOSUB88                                          :rem 172
83 PRINT"{HOME}"TAB(13)"{8 DOWN}";                       :rem 226
84 K=W8:GOSUB88                                          :rem 173
85 PRINT"{HOME}"TAB(13)"{10 DOWN}";                        :rem 6
86 K=W7:GOSUB88                                          :rem 174
87 RETURN                                                 :rem 79
88 PRINTT$(K):RETURN                                      :rem 43
89 POKE36878,0:POKE36875,0                               :rem 222
90 IFS=0THEN30                                            :rem 74
91 FORE=1TOS                                             :rem 250
92 ONEGOTO105,103,101,99,97                              :rem 212
93 NEXTE                                                 :rem 240
94 S=0:PRINT"{HOME}{3 DOWN}"TAB(6)" ":PRINT:PRINT:PRINTTAB(6)
   " ":PRINT:PRINTTAB(6)" "                               :rem 71
95 PRINT:PRINTTAB(6)" ":PRINT:PRINT:PRINTTAB(6)" "        :rem 91
96 GOTO14                                                 :rem 13
```

```
97 IFW3=W5ANDW3=W7THENB=W3:GOSUB107                          :rem 4
98 GOTO93                                                   :rem 22
99 IFW1=W5ANDW1=W9THENB=W1:GOSUB107                          :rem 2
100 GOTO93                                                  :rem 54
101 IFW3=W6ANDW3=W9THENB=W3:GOSUB107                        :rem 41
102 GOTO93                                                  :rem 56
103 IFW1=W4ANDW1=W7THENB=W1:GOSUB107                        :rem 33
104 GOTO93                                                  :rem 58
105 IFW2=W5ANDW2=W8THENB=W2:GOSUB107                        :rem 40
106 GOTO93                                                  :rem 60
107 ONBGOTO116,117,118,119,120,121,122                     :rem 218
108 PRINT"{HOME}{20 DOWN}"TAB(7)"YOU WIN"                    :rem 95
109 FORH=1TOB                                                :rem 28
110 POKE36878,15:POKE36875,219                             :rem 161
111 P=P+5                                                  :rem 208
112 PRINT"{HOME}{14 DOWN}"TAB(12)"{5 SPACES}{5 LEFT}"P
                                                           :rem 150
113 POKE36878,0:POKE36875,0                                  :rem 2
114 FORD=1TO10:NEXTD:NEXTH                                 :rem 177
115 RETURN                                                 :rem 119
116 B=3:GOTO108                                              :rem 86
117 B=10:GOTO108                                           :rem 133
118 B=20:GOTO108                                           :rem 135
119 B=50:GOTO108                                           :rem 139
120 B=100:GOTO108                                          :rem 175
121 B=500:GOTO108                                          :rem 180
122 B=1000:GOTO108                                         :rem 225
123 PRINT"{CLR}{5 SPACES}{6 UP}POT= 0"                     :rem 192
124 PRINT"{4 DOWN}{4 SPACES}PLAY AGAIN? {RVS}Y{OFF}{LEFT}";:G
    ETY$:IFY$=""THEN123                                    :rem 121
125 IFY$="Y"THEN3                                          :rem 224
126 END                                                    :rem 112
```

Linton S. Chastain

Code Game

This game will help you understand the difficulty in decoding a code that is generated from capital letters and the numbers 0–9. The program also helps you understand the subtle processes that the mind goes through to solve a problem.

The program initially requests you to input a code. It will be used by the program to generate a pseudorandom code which you'll attempt to solve. The code that you input is limited to capital letters and the numbers 0–9. The length of your code is also limited to ten characters.

Line 50 sends the computer to a subroutine that determines the length of the pseudorandom code. Lines 60–80 make sure that your input agrees with the input criteria. Line 90 sends the computer to a subroutine at lines 150–220. This subroutine generates the pseudorandom code that you are to solve.

Line 100 sends the computer to the subroutine at lines 230–380. This subroutine compares your subsequent inputs with the pseudorandom code. It gives you feedback on the correctness of your input as compared with the pseudorandom code. The subroutine also counts the number of your attempts (T), and it gives you the correct character after five attempts. The character that you are given is the first character, followed by the next characters after each of five attempts.

If you solve the code before you are given the last character, you will be told the number of attempts it took you to solve the code (T); otherwise, you will be prompted with GAME OVER CODE WAS....

Code Game

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
20 PRINT"{CLR}":B=0:T=0:E=0:R$(1)=""                    :rem 73
30 PRINT"INPUT A CODE (A->Z AND OR 0->9)"              :rem 191
35 PRINT"CODE LENGTH OF LESS":PRINT"THAN 10"           :rem 121
40 INPUTR$(0):H=LEN(R$(0)):IFH>10 THEN20                :rem 18
50 IFT<1 THEN GOSUB130                                 :rem 248
60 FORX=1TOA:FORY=58TO64:IFMID$(R$(0),X,1)=CHR$(Y) THEN20
                                                       :rem 145
70 IFASC(MID$(R$(0),X,1))<48 OR ASC(MID$(R$(0),X,1))>90 THEN2
   0                                                    :rem 21
80 NEXTY:NEXTX                                         :rem 209
90 IFT<1 THEN GOSUB150                                 :rem 254
100 GOSUB230                                           :rem 166
110 POKE209,162:POKE210,31:PRINT"DO YOU WANT TO TRY"   :rem 36
115 POKE209,184:POKE210,31:PRINT"AGAIN (Y=YES OR N=NO)"
                                                       :rem 194
116 INPUTR$                                            :rem 158
```

```
120 IFLEFT$(R$,1)="Y" THEN 20                                  :rem 0
125 IFLEFT$(R$,1)<>"Y" THEN END                               :rem 183
130 A=LEN(R$(0))                                               :rem 57
140 RETURN                                                    :rem 117
150 FORY=1TOA                                                  :rem 40
160 D(Y)=INT(RND(1)*45)+50                                    :rem 135
170 IFD(Y)>90 OR D(Y)<48 THEN160                              :rem 185
175 IFD(Y)>57 AND D(Y)<65 THEN 160                            :rem 242
180 NEXTY                                                      :rem 49
190 FORY=1TOA                                                  :rem 44
200 R$(1)=LEFT$(R$(1),Y)+CHR$(D(Y))                            :rem 79
210 NEXTY                                                      :rem 43
220 RETURN                                                    :rem 116
230 PRINT"{CLR}":FORX=1TOA                                    :rem 196
240 IFMID$(R$(0),X,1)=MID$(R$(1),X,1) THEN B=B+1              :rem 253
250 NEXTX                                                      :rem 46
255 IFB=A AND C<A THENPOKE209,66:POKE210,30:PRINT"YOU HAVE SO
    LVED THE"                                                 :rem 116
260 IFB=A AND C<A THENPOKE209,88:POKE210,30:PRINT"CODE WITH";
    T;"ATTEMPTS"                                              :rem 124
270 IFB/A<1 THEN POKE209,88:POKE210,30                        :rem 103
275 IFB/A<1 THEN PRINT INT(B/A*100);"{3 SPACES}% CORRECT"
                                                              :rem 103
280 IFB=A THEN C=0                                            :rem 200
290 IFB<>A THEN310                                            :rem 234
300 RETURN                                                    :rem 115
310 POKE209,110:POKE210,30:T=T+1:E=E+1:INPUT"TRY AGAIN";R$(0)
                                                               :rem 30
315 FORX=1TOA                                                  :rem 42
316 Z(X)=PEEK(7800+X)                                         :rem 162
317 IFZ(X)<27 THEN R$(0)=R$(0)+CHR$(Z(X)+64)                   :rem 65
318 IFZ(X)>47 AND Z(X)<58 THEN R$(0)=R$(0)+CHR$(Z(X))         :rem 48
319 NEXTX                                                      :rem 52
320 IFE=5 THENC=C+1                                            :rem 41
330 IFE=5 THEN POKE209,162:POKE210,31                          :rem 38
335 IFE=5 THENPRINTLEFT$(R$(1),C)                             :rem 164
340 IFE=5 THENE=0                                             :rem 190
345 IFC=A THEN POKE209,110:POKE210,30                          :rem 46
350 IFC=A THEN PRINT"GAME OVER CODE WAS"                       :rem 68
355 IFC=A THEN POKE209,132:POKE210,30:PRINTR$(1)              :rem 242
360 B=0:FORX=1TO1000:NEXTX                                    :rem 100
370 IFC<A THEN230                                             :rem 173
380 C=0:T=0:RETURN                                             :rem 96
390 END                                                       :rem 115
```

Stephen Hust

Logicolor

This game of logic for the unexpanded VIC is easy to play but difficult to master. The objective is deceptively simple: Guess the color pattern that the computer has preselected using the clues given after each guess.

There are four positions to match. The correct solution requires that you guess the correct color for each position. After each guess your VIC will tell how many of the colors you guessed match in the correct position, how many match but were in the wrong position, and how many don't match. Use the clues to help find the correct answer. You'll get only six tries to figure out the solution, and once you enter a guess there's no turning back.

Logicolor

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 PRINT"{CLR}":IFPEEK(44)=18GOTO15                          :rem 133
10 SC=7680:CL=38400:GOTO20                                  :rem 227
15 SC=4096:CL=37888                                          :rem 36
20 SB=36879:VL=36878:S=36874:T=36875:U=36876                :rem 246
30 POKESB,152:POKEVL,15                                     :rem 136
40 PRINT"{CLR}":PRINT:PRINT:PRINTSPC(6)"{BLK}LOGICOLOR":rem 8
45 PRINT:PRINT:PRINTSPC(2)"YOU HAVE SIX TRIES":PRINTSPC(3)"TO
    GET THE RIGHT"                                          :rem 36
46 PRINTSPC(2)"COLOR COMBINATION"                           :rem 86
50 PRINT:PRINT"'";CHR$(113);"'= ONE COLOR IS IN":PRINT"THE CO
   RRECT POSITION"                                          :rem 48
55 PRINT:PRINT"'{WHT}";CHR$(113);"{BLK}'= ONE COLOR IS IN":PR
   INT"THE WRONG POSITION"                                  :rem 69
60 PRINT:PRINTSPC(1)"{RED}CAUTION! YOU CAN'T":PRINTSPC(1)"CHA
   NGE A COLOR ONCE"                                        :rem 254
61 PRINTSPC(1)"A KEY IS STRUCK!{BLK}"                       :rem 243
65 PRINT:PRINTSPC(5)"HIT ANY KEY"                            :rem 37
70 GETB$:IFB$=""THEN70                                      :rem 243
100 A=INT(RND(1)*6)+3                                       :rem 111
110 B=INT(RND(1)*6)+3                                       :rem 113
120 C=INT(RND(1)*6)+3                                       :rem 115
130 D=INT(RND(1)*6)+3                                       :rem 117
150 RP=81:R=87:Q=63:E=160:N=200:DI=49                        :rem 98
160 RCP=0:WCP=0                                             :rem 122
170 X=SC+3*22:V=CL+3*22:P=9                                 :rem 247
200 PRINT"{CLR}"                                            :rem 246
210 PRINT"{HOME}COLOR{2 SPACES}{RED}{RVS} {OFF} {CYN}{RVS}
    {OFF} {PUR}{RVS} {OFF} {GRN}{RVS} {OFF} {BLU}{RVS} {OFF}
    {SPACE}{YEL}{RVS} {OFF}{BLK}"                           :rem 144
220 PRINT" KEY{3 SPACES}3 4 5 6 7 8"                        :rem 143
300 FORWW=1TO6                                              :rem 111
500 POKEX+3,R:POKEX+4,R:POKEX+25,R:POKEX+26,R:POKEX,DI
                                                            :rem 179
```

```
510 POKEV+3,0:POKEV+4,0:POKEV+25,0:POKEV+26,0:POKEV,0:rem 197
520 POKEX+P,Q:POKEX+P+2,E:POKEX+P+4,E:POKEX+P+6,E:POKEV+P,0
                                                    :rem 208
530 GOSUB1000                                       :rem 217
550 H=VAL(A$)                                       :rem 184
560 P=P+2                                           :rem 213
570 POKEX+P,Q:POKEV+P,0                             :rem 177
580 GOSUB1000                                       :rem 222
590 I=VAL(A$)                                       :rem 189
600 P=P+2                                           :rem 208
610 POKEX+P,Q:POKEV+P,0                             :rem 172
620 GOSUB1000                                       :rem 217
630 J=VAL(A$)                                       :rem 185
640 P=P+2                                           :rem 212
660 POKEX+P,Q:POKEV+P,0                             :rem 177
670 GOSUB1000                                       :rem 222
680 K=VAL(A$)                                       :rem 191
700 GOSUB3000                                       :rem 218
710 Z=INT(RND(1)*4)+3                               :rem 141
740 IFZ=5THENZ=25                                    :rem 35
750 IFZ=6THENZ=26                                    :rem 38
770 IFRCP=0THEN860                                   :rem 76
780 FORQL=1TORCP                                     :rem 25
790 POKEX+Z,RP:POKEV+Z,0:POKES,N:FORZZ=1TO100:NEXTZZ:POKES,0
                                                     :rem 12
800 Z=Z+1                                           :rem 229
810 IFZ=5THENZ=25                                    :rem 33
830 IFZ=27THENZ=3                                    :rem 35
840 NEXTQL                                          :rem 120
850 IFRCP=4THEN4030                                 :rem 120
860 IFWCP=0THEN940                                   :rem 80
870 FORLQ=1TOWCP                                     :rem 30
880 POKEX+Z,RP:POKEV+Z,1:POKEU,N:FORZZ=1TO100:NEXTZZ:POKEU,0
                                                     :rem 17
890 Z=Z+1                                           :rem 238
900 IFZ=5THENZ=25                                    :rem 33
920 IFZ=27THENZ=3                                    :rem 35
930 NEXTLQ                                          :rem 120
940 RCP=0:WCP=0                                     :rem 128
950 X=X+3*22:V=V+3*22:P=9:DI=DI+1                   :rem 115
960 NEXTWW                                          :rem 140
970 GOTO5000                                        :rem 158
1000 GETA$:IFA$=""THEN1000                          :rem 165
1010 IFA$>"8"ORA$<"3"THEN1000                        :rem 25
1020 POKEX+P,E:POKEV+P,VAL(A$)-1                    :rem 147
1030 RETURN                                         :rem 164
3000 IFH=ATHEN3045                                   :rem 19
3010 IFH=BANDI<>BTHEN3040                           :rem 232
3011 IFH=CANDJ<>CTHEN3040                           :rem 236
3012 IFH=DANDK<>DTHEN3040                           :rem 240
3030 GOTO3050                                       :rem 199
```

```
3040 WCP=WCP+1                                  :rem 52
3041 GOTO3050                                   :rem 201
3045 RCP=RCP+1                                  :rem 47
3050 IFI=BTHEN3085                              :rem 30
3055 IFI=AANDH<>ATHEN3080                       :rem 243
3056 IFI<>CORJ=CTHEN3060                        :rem 198
3057 IFH<>CTHEN3080                             :rem 93
3058 IFH=ATHEN3080                              :rem 31
3059 IFH<>AANDH=BTHEN3080                       :rem 247
3060 IFI<>DORK=DTHEN3070                        :rem 197
3061 IFH<>DTHEN3080                             :rem 89
3062 IFH=ATHEN3080                              :rem 26
3063 IFH<>AANDH=BTHEN3080                       :rem 242
3064 IFH<>AANDJ<>CANDH=CTHEN3080                :rem 206
3070 GOTO3100                                   :rem 199
3080 WCP=WCP+1                                  :rem 56
3081 GOTO3100                                   :rem 201
3085 RCP=RCP+1                                  :rem 51
3100 IFJ=CTHEN3135                              :rem 24
3105 IFJ=AANDH<>AANDI<>ATHEN3130                :rem 195
3106 IFJ=AANDH<>AANDI=BTHEN3130                 :rem 136
3107 IFJ=BANDI<>BANDH<>BTHEN3130                :rem 200
3108 IFJ=BANDI<>BANDH=ATHEN3130                 :rem 139
3109 IFJ<>DORK=DTHEN3129                        :rem 207
3110 IFH<>DANDI<>DTHEN3130                      :rem 42
3111 IFH=AANDI=BTHEN3130                        :rem 172
3112 IFH=AANDI=CTHEN3130                        :rem 174
3113 IFH=BANDI=CTHEN3130                        :rem 176
3114 IFH=CANDI=BTHEN3130                        :rem 177
3115 IFH=CANDI=ATHEN3130                        :rem 177
3116 IFH=BANDI=ATHEN3130                        :rem 177
3117 IFH=AANDI<>DTHEN3130                       :rem 241
3118 IFH=BANDI<>DTHEN3130                       :rem 243
3119 IFH=CANDI<>DTHEN3130                       :rem 245
3120 IFH<>DANDI=ATHEN3130                       :rem 235
3121 IFH<>DANDI=BTHEN3130                       :rem 237
3122 IFH<>DANDI=CTHEN3130                       :rem 239
3129 GOTO3150                                   :rem 209
3130 WCP=WCP+1                                  :rem 52
3131 GOTO3150                                   :rem 202
3135 RCP=RCP+1                                  :rem 47
3150 IFK=DTHEN3185                              :rem 36
3155 IFK<>AORH=ATHEN3160                        :rem 195
3156 IFI<>AANDJ<>ATHEN3180                      :rem 53
3157 IFI=BANDJ<>ATHEN3180                       :rem 250
3158 IFI=BANDJ=CTHEN3180                        :rem 192
3159 IFI<>AANDJ=CTHEN3180                       :rem 253
3160 IFK<>BORI=BTHEN3166                        :rem 200
3161 IFH<>BANDJ<>BTHEN3180                      :rem 50
3162 IFH=AANDJ<>BTHEN3180                       :rem 245
```

```
3163 IFH=AANDJ=CTHEN3180                                  :rem 186
3164 IFH<>BANDJ=CTHEN3180                                 :rem 249
3165 IFH<>BANDJ=ATHEN3180                                 :rem 248
3166 IFK<>CORJ=CTHEN3179                                  :rem 213
3167 IFH<>CANDI<>CTHEN3180                                 :rem 57
3168 IFH=AANDI=BTHEN3180                                  :rem 189
3169 IFH=AANDI<>CTHEN3180                                 :rem 252
3170 IFH=BANDI=ATHEN3180                                  :rem 182
3171 IFH=BANDI<>CTHEN3180                                 :rem 246
3172 IFH<>CANDI=ATHEN3180                                 :rem 246
3173 IFH<>CANDI=BTHEN3180                                 :rem 248
3179 GOTO4000                                             :rem 209
3180 WCP=WCP+1                                             :rem 57
3181 GOTO4000                                             :rem 202
3185 RCP=RCP+1                                             :rem 52
4000 RETURN                                               :rem 164
4030 READPP                                               :rem 131
4050 IFPP=-1THEN4120                                      :rem 137
4060 READDD                                               :rem 110
4070 POKES,PP:POKET,PP:POKEU,PP                            :rem 44
4080 FORNN=1TODD:NEXTNN                                   :rem 253
4090 POKES,0:POKET,0:POKEU,0                              :rem 222
4100 GOTO4030                                             :rem 197
4120 POKES,0:POKET,0:POKEU,0                              :rem 216
4140 RESTORE:GOTO4200                                      :rem 38
4150 DATA209,100,209,50,209,50,209,50                      :rem 68
4160 DATA219,100,209,100,225,100,232,500                  :rem 200
4190 DATA-1                                                :rem 70
4200 FORY=1TO18:PRINT:NEXT:PRINT"YOU DID IT IN";DI-48;"TRIES!
     "                                                     :rem 85
4210 GOTO5010                                             :rem 198
5000 POKES,N:POKET,N:POKEU,N:FORZZ=1TO700:NEXT             :rem 38
5005 POKES,0:POKET,0:POKEU,0:FORY=1TO20:PRINT:NEXT          :rem 8
5010 X=SC+484:V=CL+484:P=9                                :rem 227
5020 PRINT"SOLUTION";                                      :rem 80
5030 POKEX+P,E:POKEV+P,A-1                                 :rem 64
5040 P=P+2                                                  :rem 3
5050 POKEX+P,E:POKEV+P,B-1                                 :rem 67
5060 P=P+2                                                  :rem 5
5070 POKEX+P,E:POKEV+P,C-1                                 :rem 70
5080 P=P+2                                                  :rem 7
5090 POKEX+P,E:POKEV+P,D-1                                 :rem 73
5100 GETB$:IFB$=""THEN5100                                :rem 177
5110 GOTO100                                              :rem 145
```

Bryan S. Jones

Maneuver

Are you just a little weary of shooting down alien craft, dodging bombs, and playing all those cloned and recloned action games? This game will stimulate the gray matter and give quiet satisfaction if you manage to maneuver things to a successful conclusion.

This isn't a game of pure chance or mechanical skill laced with random numbers or happenings, but one where you have full control. You have all the data before you, and only your mind and cleverness can you resolve the situation. This game is similar to a card game conceived by Martin Gardner.

Thinking and Planning

The only random aspect of the game occurs when the 52 shuffled cards are laid out face up, left to right, a row at a time, in eight columns. From then on it's up to you. The first four columns each contain 7 cards, and the last four columns contain 6. This is called the board. The object of the game is to maneuver these cards into numeric order (ace to king) into the four piles above the board, one for each suit.

Below the board are four cells called temporary cells. Only one card at a time can occupy a temporary cell. Thus, a maximum of only four cards can be held in these cells at any one time.

Only cards from the bottom of each column and those in temporary cells can be moved, and they can be moved only one at a time. When a card is taken from a board column, the next card in that column becomes accessible.

Moving Cards

To pick up a card, use the joystick to place the green rectangle completely over the desired card and press the fire button. The green rectangle will disappear leaving the card visible again. The card can now be maneuvered to the desired location by using the joystick. When the card is in place, press the fire button again. The green rectangle will reappear over the card, and it is then free to move to the next card to be picked up.

When the fire button is pressed, the green rectangle will either disappear or reappear depending on the situation, and a beep will sound indicating that a legal move has been made. However, if you have made a mistake or are trying to cheat, a berating tone will be heard and the screen will remain unchanged.

There are only three places a card can be moved to. A card may be placed in its proper pile above the board, but only if it's the next card in numeric order. It may be placed in one of the temporary cells providing that it is vacant. Or it may be placed at the bottom of one of the board columns, but only if the card currently at the bottom is the next card of the same suit

(for example, a four of hearts can be placed on the five of hearts). If one of the board columns becomes empty, then any accessible card can be placed there as a new starting card.

Make sure that when you place a card in one of the four piles above the board that the suit of the card is placed over the suit which is already printed there.

When the program is running, it will take about 40 seconds to "shuffle" the cards. It will then lay out the cards for you. Unfortunately, since the VIC-20 has only 22 print positions across, the last two columns are squeezed together without an intervening space. But this does not detract from the mental challenge of the game.

The program constantly scans to see if kings top the four piles above the board. When you achieve your goal, the computer will immediately congratulate you and begin shuffling the cards for another game. If you find yourself hopelessly stuck, press any key on the keyboard, and the cards will be shuffled for another game.

Maneuver

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 DIMC%(52):POKE36878,15:PRINT"{CLR}"                    :rem 212
10 FORX=1TO52:C%(X)=X:NEXTX                              :rem 201
50 PRINT"{CLR}{4 SPACES}***MANEUVER***":PRINT            :rem 239
60 PRINT"{4 SPACES}{BLK}X{4 SPACES}{RED}Z{4 SPACES}S
   {4 SPACES}{BLK}A"                                     :rem 185
70 PRINT                                                 :rem 244
80 FORX=1TO52:A=INT(RND(1)*(53-X)+X):B%=C%(X):C%(X)=C%(A):C%(
   A)=B%                                                 :rem 173
85 PRINT"{BLK}";:IFC%(X)>26THENPRINT"{RED}";             :rem 224
90 S=INT((C%(X)-1)/13)+1:N=C%(X)-13*(S-1)                :rem 119
100 IFS=1THENS$="X "                                     :rem 224
110 IFS=2THENS$="A "                                     :rem 203
120 IFS=3THENS$="Z "                                     :rem 230
130 IFS=4THENS$="S "                                     :rem 225
140 N$=RIGHT$(STR$(N),1)                                  :rem 80
160 IFN=1THENN$="A"                                       :rem 69
170 IFN=10THENN$="T"                                     :rem 137
180 IFN=11THENN$="J"                                     :rem 129
190 IFN=12THENN$="Q"                                     :rem 138
200 IFN=13THENN$="K"                                     :rem 125
210 PRINTN$+S$;:IF(X+1)/8=INT((X+1)/8)THENPRINT"{LEFT}";
                                                          :rem 97
215 IFX/8=INT(X/8)THENPRINT"{LEFT}";                     :rem 246
220 NEXTX:FORX=1TO11:PRINT:NEXTX                         :rem 175
230 PRINT"{RVS}{BLU}{3 RIGHT}{2 SPACES}{3 RIGHT}{2 SPACES}
    {3 RIGHT}{2 SPACES}{3 RIGHT}{2 SPACES}"              :rem 243
240 PX=0:PY=12                                            :rem 46
245 H1=160:H2=160:HC1=5:HC2=5                             :rem 65
```

```
250 ZS=7680+22*PY+PX:ZC=38400+22*PY+PX                    :rem 211
260 S1=PEEK(ZS):S2=PEEK(ZS+1):C1=PEEK(ZC):C2=PEEK(ZC+1)
                                                           :rem 80
270 POKEZS,H1:POKEZS+1,H2:POKEZC,HC1:POKEZC+1,HC2        :rem 107
280 GOSUB370:IFFB=1THEN430                                 :rem 57
283 GETA$:IFA$<>""THEN820                                :rem 151
285 IFPEEK(7727)=11ANDPEEK(7732)=11ANDPEEK(7737)=11ANDPEEK(77
    42)=11THEN840                                          :rem 28
290 IFXD=0ANDYD=0THEN280                                 :rem 217
300 POKEZS,S1:POKEZS+1,S2:POKEZC,C1:POKEZC+1,C2          :rem 235
310 PX=PX+XD:PY=PY+YD                                    :rem 121
320 IFPX<0THENPX=0                                       :rem 124
330 IFPX>20THENPX=20                                     :rem 227
340 IFPY<2THENPY=2                                       :rem 132
350 IFPY>21THENPY=21                                     :rem 233
360 GOTO250                                              :rem 105
370 POKE37154,127:J%=PEEK(37152)AND128:POKE37154,255     :rem 148
380 J%=J%OR(PEEK(37137)AND127)                           :rem 144
390 XD=SGN(J%AND16)-SGN(J%AND128)                        :rem 154
400 YD=SGN(J%AND4)-SGN(J%AND8)                           :rem 253
410 FB=1-SGN(J%AND32)                                    :rem 152
420 RETURN                                               :rem 118
430 IFHC1=5THEN700                                         :rem 26
440 IFPY=21ANDS1=160ANDS2=160THEN680                     :rem 148
445 IFPY=21THEN660                                         :rem 64
450 IFPY<>2THEN550                                         :rem 70
460 IFS2<>H2THEN660                                      :rem 109
470 IFS1=32ANDH1=1THEN680                                :rem 215
480 IFS1=1ANDH1=50THEN680                                :rem 216
490 IFS1=57ANDH1=20THEN680                                 :rem 17
500 IFS1=20ANDH1=10THEN680                               :rem 254
510 IFS1=10ANDH1=17THEN680                                  :rem 5
520 IFS1=17ANDH1=11THEN680                                  :rem 7
530 IFS1=>50ANDS1<=56ANDS1+1=H1THEN680                   :rem 252
540 GOTO660                                              :rem 110
550 IFS1<>32ORS2<>32THEN660                                :rem 92
560 IFPY<4THEN660                                          :rem 14
570 IFINT(PX/3)<>PX/3ANDPX<>20THEN660                      :rem 23
580 IFPY=4THEN680                                          :rem 19
590 IFPEEK(ZS-21)<>H2THEN660                             :rem 159
600 G=PEEK(ZS-22):IFG=11ANDH1=17THEN680                    :rem 59
610 IFG=17ANDH1=10THEN680                                :rem 201
620 IFG=10ANDH1=20THEN680                                :rem 196
630 IFG=50ANDH1=1THEN680                                 :rem 152
640 IFG=20ANDH1=57THEN680                                :rem 209
650 IFG=>51ANDG<=57ANDG=H1+1THEN680                        :rem 74
660 POKE36874,160:FORC=1TO150:NEXT:POKE36874,150           :rem 85
670 FORC=1TO150:NEXT:POKE36874,0:GOTO280                 :rem 197
680 POKE36876,235:S1=H1:S2=H2:C1=HC1:C2=HC2              :rem 219
690 H1=160:H2=160:HC1=5:HC2=5:FORC=1TO150:NEXT:POKE36876,0:GO
    TO270                                                :rem 168
```

```
700 IFPY<4THEN660                                        :rem 10
710 IFPY<>21THEN770                                     :rem 122
720 IFNOT(PX=3ORPX=8ORPX=13ORPX=18)THEN660              :rem 228
730 IFC1=6ORC2=6THEN660                                 :rem 100
740 POKE36876,235:H1=S1:H2=S2:HC1=C1:HC2=C2             :rem 216
750 S1=160:S2=160:C1=6:C2=6:FORC=1TO150:NEXT             :rem 78
760 POKE36876,0:GOTO270                                  :rem 66
770 IFNOT(S2=65ORS2=83ORS2=88ORS2=90)THEN660            :rem 212
780 IFPEEK(ZS+22)<>32ORPEEK(ZS+23)<>32THEN660           :rem 189
790 POKE36876,235:H1=S1:H2=S2:HC1=C1:HC2=C2             :rem 221
800 S1=32:S2=32:C1=1:C2=1:FORC=1TO150:NEXT              :rem 220
810 POKE36876,0:GOTO270                                  :rem 62
820 PRINT"{CLR}";SPC(224);"YOU HAVE LOST!"               :rem 11
830 PRINT"{DOWN}YOU NEED TO PRACTICE!":FORX=1TO2000:NEXT:GOTO
    50                                                   :rem 66
840 PRINT"{CLR}";SPC(224);"CONGRATULATIONS!"             :rem 39
850 PRINTSPC(26);"YOU HAVE WON!!"                       :rem 159
860 FOR X=1 TO 3000:GOTO50                              :rem 137
```

Bill
McDannell

Battleship

"Battleship" is a version of the traditional game in two parts for the unexpanded VIC and two players.

The first program introduces this traditional game to the newcomer. The second program must be saved using the filename BSHIP.PRG. Tape users should delete lines 950 and 960 of Program 1, and change line 945 to

945 POKE 198,1:POKE 631,131

Each player starts with four ships hidden on the grid: A battleship is five squares long, a cruiser occupies four squares, a destroyer is three squares, and a submarine is two. The ships may be located vertically, horizontally, or in either diagonal direction. You begin by being permitted to fire five volleys per round. Each time your opponent sinks one of your ships, you get one less volley per round. The first player to sink his or her opponent's entire fleet is the winner.

This game differs just a bit from the paper-and-pencil version since the computer sets up both battlefields. This adds an extra element of excitement to the traditional game. Not only do you not know where your opponent's fleet is hidden, but you don't know where yours is either, so there's considerable suspense as your opponent takes shots. And the computer can lay out some very sneaky battlefields.

Program 1. Battleship

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
2 GOSUB500                                          :rem 71
5 X=8174:Y=30720:Z=500                             :rem 208
10 V=36878:N=36877:S=36874:S1=36875                :rem 182
20 POKEV,15                                        :rem 121
30 POKES,128:POKES1,170                             :rem 93
35 FORT=1TO700:NEXTT                                :rem 24
40 POKES,128:POKES1,130                             :rem 90
50 FORM=1TO900:NEXTM                                 :rem 9
60 FORB=15TO0STEP-.02                              :rem 254
61 POKEV,B                                          :rem 90
62 NEXTB                                           :rem 233
70 POKES,0:POKES1,0                                :rem 142
80 FORT=500TO0STEP-1:POKEV,T/100:POKEN,200          :rem 86
90 IFT=ZTHENGOSUB200                                :rem 36
95 IFT<10THENPOKEN,0:GOTO600                       :rem 255
100 NEXTT:POKEN,0:END                               :rem 72
200 POKEX,93:POKEX+Y,0:Z=Z-50                       :rem 15
210 POKEX+22,32:X=X-22                             :rem 242
220 RETURN                                         :rem 116
500 PRINT"{CLR}{9 DOWN}{10 SPACES}[2 F]"             :rem 8
```

```
510 PRINT"{9 SPACES}{RVS}£{2 SPACES}[*]{OFF}"          :rem 147
520 PRINT"[8 @]{RVS}{4 SPACES}{OFF}£[8 @]":RETURN:rem 238
600 POKE36879,42                                       :rem 104
610 PRINT"{WHT}{CLR}{8 DOWN}"                          :rem 136
620 PRINT"{5 SPACES}{RVS}£[*]£[*]£[*]£[*]£[*]£[*]{OFF}"
                                                        :rem 61
630 PRINT"{4 SPACES}{RVS}£{12 SPACES}[*]{OFF}"         :rem 150
640 PRINT"{4 SPACES}[*]{RVS} BATTLESHIP! {OFF}£"       :rem 168
650 PRINT"{4 SPACES}{RVS}£{12 SPACES}[*]{OFF}"         :rem 152
660 PRINT"{5 SPACES}[*]£[*]£[*]£[*]£[*]£[*]£"          :rem 157
670 POKEN,220                                          :rem 218
680 FORL=15TO0STEP-1                                   :rem 225
690 POKEV,L                                            :rem 156
700 FORM=1TO300:NEXTM                                   :rem 53
710 NEXTL                                               :rem 35
720 POKEV,0:POKEN,0                                    :rem 141
750 PRINT"{2 DOWN}INSTRUCTIONS?(Y/N)                   :rem 136
760 GETA$:IFA$=""THEN760                                :rem 93
770 IFA$="N"THENPOKE36879,27:PRINT"{CLR}{BLU}":GOTO930:rem 49
780 IFA$="Y"THEN800                                     :rem 52
790 GOTO760                                            :rem 118
800 POKE36879,27:PRINT"{CLR}{BLK}"                     :rem 155
810 PRINT"BATTLESHIP! IS A GAME FOR 2 PLAYERS."        :rem 217
820 PRINT"{DOWN}EACH PLAYER HAS HIS{3 SPACES}OWN BATTLE ZONE
    {SPACE}AND A FLEET OF FOUR SHIPS."                  :rem 16
830 PRINT"{DOWN}A BATTLESHIP IS FIVE{2 SPACES}GRID SQUARES LO
    NG,                                                :rem 248
840 PRINT"{DOWN}A CRUISER IS FOUR,"                    :rem 224
850 PRINT"{DOWN}A DESTROYER IS THREE,"                 :rem 193
860 PRINT"{DOWN}AND A SUB IS TWO."                      :rem 66
870 PRINT"{2 DOWN}HIT ANY KEY"                          :rem 72
880 GETA$:IFA$=""THEN880                                :rem 99
890 PRINT"{CLR}{DOWN}EACH PLAYER BEGINS{4 SPACES}WITH VOLLEYS
     OF FIVE{2 SPACES}SHOTS."                          :rem 148
900 PRINT"{DOWN}IF YOUR OPPONENT SINKSA SHIP, YOU LOSE ONE
    {2 SPACES}SHOT."                                   :rem 115
910 PRINT"{DOWN}THE WINNER IS THE{5 SPACES}FIRST PLAYER TO SI
    NK{2 SPACES}HIS OPPONENT'S ENTIRE FLEET."           :rem 16
920 PRINT"{DOWN}FIRE BY ENTERING GRID COORDINATES (H7,B6,
    {3 SPACES}ETC.)                                    :rem 131
930 PRINT"{DOWN}WHEN YOU'RE READY TO{2 SPACES}PLAY, PRESS RET
    URN"                                                :rem 82
940 GET R$:IF R$="" THEN 940                           :rem 127
945 PRINT"{CLR}{BLK}PLEASE WAIT":POKE 198,2:POKE 631,13:POKE
    {SPACE}632,13                                       :rem 21
950 PRINT"{HOME}{WHT}":PRINT "{2 DOWN}LOAD";CHR$(34);"BSHIP.P
    RG";CHR$(34);",8"                                  :rem 171
960 PRINT"{5 DOWN}RUN":PRINT"{HOME}":END               :rem 233
```

Program 2. Bship.prg

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 POKE648,28                                                    :rem 156
10 POKE56,28:CLR                                                :rem 172
20 DIML%(23),P(5,5,2),B(2),C(2),D(2),S(2),U(5),V(4),W(3),Y(2)
   :R=7705                                                      :rem 181
30 GOSUB800:PRINT"{CLR}{BLK}":GOSUB800                          :rem 251
50 E=1:F=1                                                        :rem 6
60 PRINT"{CLR}":CO=30720:N(1)=5:N(2)=5                          :rem 200
70 ZX=ZX+1                                                      :rem 100
80 FORT=0TO9                                                    :rem 236
90 PRINTT"{RVS}{10 SPACES}{OFF}"                                 :rem 50
100 NEXTT                                                        :rem 36
110 PRINT"{3 SPACES}ABCDEFGHIJ"                                  :rem 26
120 IFZX=1THENGOSUB800:GOTO60                                    :rem 98
130 PRINT"{2 DOWN}"                                             :rem 135
140 FORZ=1TO4                                                    :rem 27
150 ONZGOTO160,170,180,190                                      :rem 172
160 L=5:GOTO200                                                  :rem 90
170 L=4:GOTO200                                                  :rem 90
180 L=3:GOTO200                                                  :rem 90
190 L=2                                                          :rem 85
200 X=INT(RND(1)*207)+R                                           :rem 9
210 IFPEEK(X)<>160THEN200                                       :rem 194
220 A=INT(RND(1)*4)+1                                           :rem 110
230 ONAGOTO240,250,260,270                                      :rem 142
240 I=1:GOTO280                                                  :rem 90
250 I=21:GOTO280                                                :rem 141
260 I=22:GOTO280                                                :rem 143
270 I=23                                                        :rem 132
280 FORM=1TOL                                                    :rem 43
290 P(M,Z,E)=X                                                   :rem 21
300 IFPEEK(X)<>160THEN200                                       :rem 194
305 IFI=1ORI=22THEN320                                           :rem 45
310 IFPEEK(X+1)<>160ANDPEEK(X+22)<>160THEN200                    :rem 96
320 POKEX,224:X=X+I                                             :rem 123
330 NEXTM                                                        :rem 34
340 NEXTZ                                                        :rem 48
350 PRINT"{HOME}"                                               :rem 124
360 FORT=1TO10                                                   :rem 70
370 IFE=2THEN390                                                :rem 168
380 PRINT"{3 RIGHT}{RVS}{BLK}LLLLLLLLLL{OFF}":GOTO400           :rem 246
390 PRINT"{3 RIGHT}{RVS}{GRN}LLLLLLLLLL{OFF}{BLU}"              :rem 157
400 NEXTT                                                        :rem 39
410 IFE=1THENE=2:R=7193:GOSUB800:GOTO130                        :rem 176
420 PRINT"{DOWN}PLAYER"F"TURN"K+1;                              :rem 250
430 K=K+1                                                       :rem 198
435 POKE198,0                                                   :rem 201
440 GETA$:IFA$=""THEN440                                         :rem 83
450 IFASC(A$)<65ORASC(A$)>74THEN440                             :rem 250
460 PRINTA$;:GOTO480                                            :rem 214
```

```
470 GOTO440                                                    :rem 108
480 GETB$:IFB$=""THEN480                                        :rem 93
490 IFASC(B$)<48ORASC(B$)>57THEN480                              :rem 6
500 PRINTB$:A=VAL(B$):GOTO520                                  :rem 228
510 GOT0480                                                     :rem 76
520 G=ASC(A$)-65                                                :rem 64
530 H=R+G+(22*A)                                                :rem 44
540 FORT=1TO5                                                   :rem 26
550 IFH=P(T,1,E)THENP(T,1,E)=0:Q=2:CC=2:U(E)=U(E)+1:GOTO610
                                                               :rem 172
555 IFPEEK(H)=2THEN620                                          :rem 34
560 IFH=P(T,2,E)THENP(T,2,E)=0:Q=3:CC=7:V(E)=V(E)+1:GOTO610
                                                               :rem 183
565 IFPEEK(H)=3THEN620                                          :rem 36
570 IFH=P(T,3,E)THENP(T,3,E)=0:Q=4:CC=5:W(E)=W(E)+1:GOTO610
                                                               :rem 187
575 IFPEEK(H)=4THEN620                                          :rem 38
580 IFH=P(T,4,E)THENP(T,4,E)=0:Q=19:CC=0:Y(E)=Y(E)+1:GOTO610
                                                               :rem 243
585 IFPEEK(H)=19THEN620                                         :rem 93
590 NEXTT                                                       :rem 49
600 POKEH,204:POKEH+CO,3:GOTO620                               :rem 167
610 GOSUB1200:POKEH,Q:POKEH+CO,CC                               :rem 40
620 IFU(E)=5ANDB(E)=0THENN(F)=N(F)-1:B(E)=1                    :rem 176
630 IFV(E)=4ANDC(E)=0THENN(F)=N(F)-1:C(E)=1                    :rem 179
640 IFW(E)=3ANDD(E)=0THENN(F)=N(F)-1:D(E)=1                    :rem 182
650 IFY(E)=2ANDS(E)=0THENN(F)=N(F)-1:S(E)=1                    :rem 214
660 IFB(E)+C(E)+D(E)+S(E)=4THEN1000                            :rem 129
680 IFK=N(E)ANDR=7193THENR=7705:GOTO700                        :rem 106
690 IFK=N(E)ANDR=7705THENR=7193                                 :rem 97
700 IFK=N(E)THENK=0:TT=E:E=F:F=TT:FORT=1TO1000:NEXTT:GOSUB800
                                                               :rem 170
710 PRINT"{UP}PLAYER"F"TURN"K+1"{2 SPACES}{2 LEFT}";           :rem 250
720 GOTO430                                                    :rem 105
800 SW=PEEK(648)                                               :rem 151
810 IFSW=28THENSW=30:TS=150:GOTO900                            :rem 178
820 IFSW=30THENSW=28:TS=22:GOTO900                             :rem 129
830 STOP                                                       :rem 225
900 POKE648,SW:POKE36866,TS                                    :rem 137
910 FORJ=0TO23                                                  :rem 64
920 VS=PEEK(J+217):POKEJ+217,L%(J)                             :rem 182
930 L%(J)=VS                                                   :rem 142
940 NEXTJ                                                       :rem 38
950 RETURN                                                     :rem 126
1000 PRINT"{2 DOWN}GAME OVER,PLAYER"F"WON":END                 :rem 146
1200 POKE36878,10                                              :rem 143
1210 FORT=230TO128STEP-1:POKE36876,T                           :rem 160
1220 NEXTT                                                      :rem 88
1230 POKE36876,0:POKE36877,200                                 :rem 149
1240 FORT=15TO0STEP-.05                                        :rem 116
1250 POKE36878,T:NEXTT                                          :rem 84
1260 POKE36877,0:RETURN                                        :rem 125
```

Diana N.
Rowland

Flipper

If you're looking for a challenge to your mental dexterity and a change of pace from bombs and gobblers, then "Flipper," for the unexpanded VIC, is for you. You can play against yourself or against a friend. Play a round once or replay each round until you've mastered it—or until you've given up. Simple but devilish!

The screen presents you with a list of nine numbers and asks, "WHICH PAIR TO FLIP?" Your challenge is to get the list in numeric order, putting the lowest in the first position and the highest in the ninth position, by flipping as few pairs as possible. Once the numbers are in order, the computer will tell you how many turns you took and how that score compares with other turns, if any, with the same set of numbers. You're also offered the option of repeating the same set of numbers as often as you wish or of going on to a new set anytime. When the computer asks, "TRY AGAIN WITH THE SAME NUMBERS?" type in either Y or N and hit RETURN. Start the program with RUN.

Flipper

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
100 DIMM(10):DIMW(9)                                        :rem 95
110 PRINT"{CLR}"                                           :rem 246
200 FORN=1TO9                                               :rem 17
210 M(N)=INT(RND(0)*9+1):W(N)=M(N)                         :rem 117
220 IFN>1ANDM(N)=M(N-1)THEN210                             :rem 232
230 NEXTN                                                   :rem 34
240 C$=""                                                  :rem 126
250 GOSUB1000                                              :rem 216
300 T=0:IFC$<>"Y"THENZ=0                                   :rem 145
310 PRINT"{HOME}":FORN=1TO4:PRINT:NEXT:PRINT,"WHICH PAIR",,,,
    ;SPC(1);"TO FLIP?":PRINT,,,                            :rem 108
320 INPUTA$:IFVAL(A$)=0ORVAL(A$)>9GOTO310                  :rem 215
330 PRINT,,,:INPUTB$:IFVAL(B$)<=9ANDVAL(B$)>=1GOTO350:rem 214
340 PRINT"{HOME}":FORT=1TO18:PRINT,:NEXT:GOTO330           :rem 223
350 A=VAL(A$):B=VAL(B$)                                      :rem 2
360 T=T+1:M(10)=M(A):M(A)=M(B):M(B)=M(10)                  :rem 187
370 POKE7685+44*A,48+M(A):POKE38405+44*A,6                  :rem 16
380 POKE7685+44*B,48+M(B):POKE38405+44*B,6                  :rem 20
400 FORP=2TO9:IFM(P)<M(P-1)GOTO310                          :rem 34
410 NEXT                                                   :rem 212
420 C$="":GOSUB1000                                        :rem 249
500 PRINT"{HOME}":FORN=1TO11:PRINT,:NEXT:PRINT"YOU WON IN",,,
    ," ";T;"TURNS"                                         :rem 248
505 IFZ=0ORZ=TTHEN525                                       :rem 71
```

```
510 IFZ>TTHENA$="BETTER"                                  :rem 236
515 IFZ<TTHENA$="WORSE "                                  :rem 185
520 PRINT,,,"YOUR SCORE",,,,"{3 SPACES}WAS",,,,ABS(Z-T);"TURN
    (S)",,,,A$;" THAN",,,"YOUR BEST"                      :rem 184
525 IFZ=TTHENPRINT,,,"IT WAS THE",,,,"SAME SCORE",,,,"AS BEFO
    RE":FORN=1TO4:PRINT:NEXT                              :rem 200
530 IFZ<>0THENPRINT,,:GOTO540                              :rem 75
535 FORN=1TO10:PRINT:NEXT:IFZ=0THENPRINT,,                 :rem 40
540 IFZ=0ORZ>TTHENZ=T                                     :rem 150
545 PRINT"TRY AGAIN WITH THE","SAME NUMBERS?"             :rem 220
550 INPUTC$:IFC$="N"THEN110                                :rem 81
560 IFC$<>"Y"THENPRINT"{HOME}":FORT=1TO19:PRINT:NEXT:GOTO550
                                                          :rem 248
565 GOSUB1000                                             :rem 225
570 GOTO300                                               :rem 104
1000 PRINT"{CLR}":FORN=1TO9:IFC$="Y"THENM(N)=W(N)          :rem 54
1010 PRINT"{HOME}";SPC(33);"{RVS}FLIPPER"                  :rem 33
1020 POKE7680+44*N,48+N:POKE38400+44*N,6                  :rem 184
1030 POKE7680+44*N+1,46:POKE38400+44*N+1,6                :rem 246
1040 PRINT"{HOME}";SPC(33);"FLIPPER"                       :rem 18
1050 POKE7685+44*N,48+M(N):POKE38405+44*N,6:NEXT          :rem 220
1060 RETURN                                               :rem 167
```

Harvey J.
Stapleton

Poker Challenge

"Poker Challenge" utilizes the excellent character set of the VIC to generate a video version of poker solitaire which can be run on the unexpanded VIC.

Depending upon player options, from 25 to 30 cards are dealt face up, one at a time, and are positioned on a five row × five column array after the input of a two-digit row/column number. The object is to produce the best ten poker hands composed of the five rows and five columns. Once placed in position, a card may not be moved, but up to 5 cards can be discarded at the time they are dealt by assigning them to row 0, column 0. When 25 cards have been played, the program reports the score and type of poker hand for each of the five rows and five columns, as well as your total score.

The points awarded the various poker hands are displayed on the screen during play, because the relative ranking differs slightly from standard poker. The scoring of each hand is calculated in a subroutine beginning at line 2200. This scoring reflects the actual odds for that game. Changes in the scoring can be made by altering lines 1200, 1210, and 2400–2480.

The program is a tight fit on the unexpanded VIC, so leave no unnecessary space. The game is a challenge for all members of the family. A score of 50 can be considered a good score.

Poker Challenge

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
11 DIM YD%(5),ZD%(5),MY%(5,5),MZ%(5,5),DT%(4,13)      :rem 180
12 PRINT"{CLR}":H=5                                   :rem 187
13 FORR=1TO5:FORC=1TO5:MY%(R,C)=0:CR=5:X1=146:X2=131:X3=176+R
   :X4=176+C:GOSUB1800:NEXTC                          :rem 238
14 NEXT R                                             :rem 246
15 GOSUB 1200                                         :rem 169
16 FORI=1TO4:FORJ=1TO13:DT%(I,J)=0:NEXTJ,I            :rem 104
17 DX%=0:XY=RND(-TI)                                  :rem 189
18 IF DX%=25 THEN 27                                  :rem 245
19 Y%=INT(RND(1)*4)+1:Z%=INT(RND(1)*13)+1               :rem 3
20 IF DT%(Y%,Z%)<>0 THEN 19                           :rem 107
21 DT%(Y%,Z%)=1                                         :rem 8
22 GOSUB 1500                                         :rem 170
23 GOSUB 1300                                         :rem 169
24 GOSUB 1800                                         :rem 175
25 MY%(R,C)=Y%:MZ%(R,C)=Z%                            :rem 211
26 GOTO 18                                             :rem 10
27 R=6:C=7:CR=1:TT=0:GOSUB 1800                        :rem 46
28 FOR KK=1TO10                                        :rem 89
29 IF KK>5 THEN 40                                    :rem 150
30 FOR KL=1TO5                                         :rem 39
31 YD%(KL)=MY%(KK,KL):ZD%(KL)=MZ%(KK,KL):NEXT KL      :rem 104
32 GOSUB 2200                                         :rem 169
```

```
33 IFKK=1THEN PRINT"{HOME}{BLK}"TAB(12-(WT<10));WT:GOTO 37
                                                    :rem 176
34 MM=3*(KK-1)-1                                     :rem 62
35 PRINT"{HOME}":FOR LI=1 TO MM:PRINT:NEXT          :rem 237
36 PRINT TAB(12-(WT<10));WT                          :rem 45
37 PRINT"{HOME}":FORL=1TO16:PRINT:NEXT              :rem 115
38 PRINT"ROW";KK;"{15 SPACES}" :PRINTA$:PRINT"HIT ANY KEY
   {10 SPACES}":GOSUB2500                           :rem 242
39 NEXT KK                                           :rem 65
40 FOR KL=1TO5                                       :rem 40
41 YD%(KL)=MY%(KL,KK-5):ZD%(KL)=MZ%(KL,KK-5):NEXT KL  :rem 45
42 GOSUB 2200                                       :rem 170
43 PRINT"{HOME}":FORLI=1TO14:PRINT:NEXT LI           :rem 76
44 PRINT TAB(-(WT<10)+3*(KK-6));WT:PRINT             :rem 98
45 PRINT"COLUMN";KK-5;{13 SPACES}"                  :rem 152
46 PRINT A$:PRINT"HIT ANY KEY{10 SPACES}":GOSUB 2500:NEXT KK
                                                    :rem 173
47 PRINT"{HOME}":FOR LI=1TO16:PRINT:NEXT            :rem 189
48 PRINT"YOUR TOTAL SCORE";TT                       :rem 111
49 PRINT"PUSH ANY KEY FOR A{2 SPACES}"              :rem 119
50 PRINT"NEW GAME{11 SPACES}":PRINTTAB(18)"{3 SPACES}"
                                                    :rem 214
51 GOSUB 2500                                       :rem 173
52 CLR:GOTO 11                                       :rem 29
1200 PRINTTAB(17)"{BLK}PTS.":PRINTTAB(17)"SF30":PRINTTAB(17)"
     4K16":PRINTTAB(17)" S12"                        :rem 66
1210 PRINTTAB(17)"FH10":PRINTTAB(17)"3K 6":PRINTTAB(17)" F 5"
                                                    :rem 121
1215 PRINTTAB(17)"2P 3":PRINTTAB(17)"1P 1"           :rem 44
1220 RETURN                                         :rem 165
1300 PRINT"{HOME}":FORJ=1TO19:PRINT:NEXT            :rem 206
1310 IFH<0THENH=0                                   :rem 236
1320 PRINTH;"DISCARDS REMAIN":PRINT" RC=00 FOR DISCARD"
                                                     :rem 65
1330 PRINT"{5 UP} YOUR NEXT CARD IS":PRINT" TO POSITION IT
     {4 SPACES}{RVS}{GRN}RC{OFF}{BLK}"               :rem 88
1340 INPUT " HIT 2 DIGIT RC #";RC$                  :rem 208
1350 IFRC$="00"THEN 1400                            :rem 230
1360 R$=LEFT$(RC$,1):R=VAL(R$)                      :rem 166
1370 C$=RIGHT$(RC$,1):C=VAL(C$)                     :rem 205
1380 IF MY%(R,C)<>0GOTO1300                         :rem 223
1390 RETURN                                         :rem 173
1400 H=H-1                                          :rem 240
1410 IFH>=0THENDX%=DX%-1:GOTO18                     :rem 112
1420 GOTO 1300                                      :rem 196
1500 DX%=DX%+1                                      :rem 225
1510 IF Y%<3THENCR=0                                :rem 116
1520 IF Y%>2 THENCR=2                               :rem 120
1530 IF Z%=1 THEN X1=129                            :rem 214
1540 IF Z%>1 AND Z%<10{5 SPACES}THEN X1=Z%+176      :rem 115
```

```
1550 IFZ%=10THENX1=152                                          :rem 4
1560 IFZ%=11THENX1=138                                         :rem 10
1570 IFZ%=12THENX1=145                                         :rem 10
1580 IFZ%=13THENX1=139                                         :rem 15
1590 X2=160:X3=160                                            :rem 198
1600 IFY%=1THENX4=216                                         :rem 211
1610 IFY%=2THENX4=193                                         :rem 217
1620 IFY%=3THENX4=218                                         :rem 217
1630 IFY%=4THENX4=211                                         :rem 212
1640 R=6:C=7:GOSUB1800                                          :rem 4
1650 RETURN                                                   :rem 172
1800 A=38423+3*(C-1)+66*(R-1):B=7703+3*(C-1)+66*(R-1)  :rem 77
1810 POKEA,CR:POKEA+1,CR:POKEA+22,CR:POKEA+23,CR      :rem 183
1820 POKEB,X1:POKEB+1,X2:POKEB+22,X3:POKEB+23,X4:RETURN
                                                              :rem 172
2200 J=0:JF=12:JS=10                                           :rem 90
2210 FOR K=1TO5                                                :rem 61
2215 FOR M=1TO5                                                :rem 68
2220 IFZD%(M)=ZD%(K) THEN J=J+1                               :rem 174
2230 NEXT M:NEXT K                                             :rem 23
2245 ON -(J=5)-2*(J=7)-3*(J=9)GOTO 2250,2410,2420     :rem 192
2247 ON -(J=11)-2*(J=13)-3*(J=17) GOTO 2440,2450,2470 :rem 85
2250 FORW=2TO5:IFYD%(W)<>YD%(1)THENJF=0                       :rem 107
2260 NEXT W                                                    :rem 96
2285 FOR K=1TO10                                              :rem 117
2290 FOR M=1TO5                                                :rem 71
2295 IF ZD%(M)=KTHEN 2310                                      :rem 63
2300 NEXT M                                                    :rem 81
2305 NEXT K                                                    :rem 84
2310 IF K=1 THEN 2340                                           :rem 6
2315 FOR M=1TO5:IFZD%(M)>K+4THEN2335                           :rem 83
2320 NEXT M                                                    :rem 83
2330 J=JS+JF:GOTO 2399                                        :rem 241
2335 J=JF:GOTO 2399                                            :rem 46
2340 TL=0                                                     :rem 214
2345 FORM=1TO5:IFZD%(M)>5THEN2365                             :rem 228
2350 NEXT M                                                    :rem 86
2360 J=JS+JF:GOTO2399                                         :rem 244
2365 FORM=1TO5:IFZD%(M)<10THENTL=TL+1                          :rem 25
2375 NEXT M                                                    :rem 93
2380 IF TL=1 THEN 2390                                        :rem 103
2385 J=JF:GOTO2399                                             :rem 51
2390 J=JS+JF:GOTO2399                                         :rem 247
2399 ON -(J=5)-2*(J=10)-3*(J=12)-4*(J=22)GOTO2400,2460,2430,2
     480                                                      :rem 226
2400 WT=0:A$="NOTHING; 00 POINTS{3 SPACES}":RETURN    :rem 167
2410 WT=1:TT=TT+1:A$="ONE PAIR; 01 POINT{3 SPACES}":RETURN
                                                              :rem 113
2420 WT=3:TT=TT+3:A$="TWO PAIR; 03 POINTS{2 SPACES}":RETURN
                                                              :rem 227
```

```
2430 WT=5:TT=TT+5:A$="FLUSH; 05 POINTS{5 SPACES}":RETURN
                                                    :rem 70
2440 WT=6:TT=TT+6:A$="3 OF A KIND; 06 PTS. ":RETURN  :rem 63
2450 WT=10:TT=TT+10:A$="FULL HOUSE; 10 POINTS":RETURN:rem 209
2460 WT=12:TT=TT+12:A$="STRAIGHT; 12 POINTS{2 SPACES}":RETURN
                                                   :rem 135
2470 WT=16:TT=TT+16:A$="4 OF A KIND; 16 PTS. ":RETURN:rem 166
2480 WT=30:TT=TT+30:A$="STR. FLUSH; 30 POINTS":RETURN:rem 204
2500 GET A$                                         :rem 12
2510 IF{2 SPACES}A$=""THEN 2500                     :rem 51
2520 RETURN                                        :rem 169
```

Chapter Seven

Mazes

John Clopton
and
Doug Thorsvik

Heist

You have just blasted through the bottom level of an underground vault. Your blast into the vault has set off a 90-second fail-safe alarm. You will have to blast walls to make your way through the different levels. If you don't make it out in time, the security guards will be upon you.

There are three levels, each with more moneybags and more alarms. To move from one level to another, exit from your present level through the opening at the bottom. If you run into an alarm or blast beside one, you'll be caught, and the money will be returned to the vault.

Use the joystick to race through the levels and collect as many $100 moneybags as possible within the time frame, but don't be too greedy.

Heist

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 PRINT"{CLR}SETTING UP..."                                  :rem 230
10 FORI=7168TO7679:POKEI,PEEK(I+25600):NEXT                   :rem 98
15 POKE36869,255:HS=0                                        :rem 175
20 READN:IFNTHENFORN=NTON+15:READA:POKEN,A:NEXT:GOTO20:rem 93
25 S=0:A=374:B=7724:C=30720:W=0:S1=5:S2=3:V=0:Q=1             :rem 19
30 PRINT"{CLR}{HOME}{RED}{RIGHT}LEVEL";Q:PRINT"{UP}{13 RIGHT}
   HS:$";HS                                                  :rem 251
35 IFW=3THEN605                                              :rem 136
40 POKE36879,29                                               :rem 59
45 GOSUB190                                                  :rem 131
50 Z=7712:FB=0                                               :rem 252
55 POKEZ,32                                                  :rem 132
60 IFJ4THEN60                                                  :rem 8
65 IFTI$>"000130"THEN385                                      :rem 48
70 IFJ1THENZ=Z+22:IFPEEK(Z)=61THENZ=Z-22                      :rem 14
75 IFPEEK(Z)=42THENZ=Z-22                                     :rem 31
80 IFJ2THENZ=Z-1:IFPEEK(Z)=61THENZ=Z+1                       :rem 170
85 IFPEEK(Z)=42THENZ=Z+1                                     :rem 235
90 IFJ3THENZ=Z-22:IFPEEK(Z)=61ORZ<7680THENZ=Z+22              :rem 30
95 IFPEEK(Z)=42THENZ=Z+22                                     :rem 31
100 IFJ0THENZ=Z+1:IFPEEK(Z)=61THENZ=Z-1                      :rem 209
105 IFPEEK(Z)=42THENZ=Z-1                                     :rem 22
110 IFPEEK(Z)=45THENGOSUB385                                 :rem 230
115 IFPEEK(Z)=44THENS=S+100:GOSUB585                         :rem 197
120 IFFBTHENGOSUB445                                         :rem 246
125 PRINT"{HOME}{21 DOWN}{14 RIGHT}";TI$                     :rem 115
130 PRINT"{HOME}{21 DOWN}{7 RIGHT}";S                         :rem 54
135 POKEZ,59:POKEZ+30720,6                                     :rem 8
140 IFZ=8108THENS1=S1+5:S2=S2+2:W=W+1:Q=Q+1:GOTO30            :rem 10
145 DD=37154:P1=37151:P2=37152                                :rem 87
150 POKEDD,127:P=PEEK(P2)AND128:J0=-(P=0)                    :rem 108
155 POKEDD,255:P=PEEK(P1)                                    :rem 216
```

```
160 J1=-((PAND8)=0)                                  :rem 230
165 J2=-((PAND16)=0)                                  :rem 27
170 J3=-((PAND4)=0)                                  :rem 229
175 FB=-((PAND32)=0)                                  :rem 38
185 GOTO55                                            :rem 65
190 FORR=0TO2:POKE7689+R,42:NEXTR                     :rem 54
195 FORR=0TO2:POKE7689+R+C,0:NEXTR                   :rem 115
200 X=7702                                           :rem 247
205 FORI=1TO22                                        :rem 60
210 POKEX,42:POKEX+C,0:X=X+1:NEXTI                   :rem 124
215 POKE7712,32                                       :rem 41
220 X=7724                                           :rem 253
225 FORI=1TO18                                        :rem 67
230 FORJ=1TO2                                          :rem 9
235 POKEX,42:POKEX+C,0:X=X-1:NEXTJ                   :rem 134
240 X=X+24:NEXTI                                     :rem 214
245 X=8098                                             :rem 9
250 FORI=1TO22                                        :rem 60
255 POKEX,42:POKEX+C,0:X=X+1:NEXTI                   :rem 133
260 POKE8108,32                                       :rem 41
265 FORR=0TO2:POKE8129+R,42:NEXTR                     :rem 47
270 FORR=0TO2:POKE8129+R+C,0:NEXTR                    :rem 99
275 FORI=1TO160                                      :rem 118
280 X=INT(RND(1)*A)+B                                :rem 169
285 IFPEEK(X)=61THEN280                              :rem 105
290 IFPEEK(X)=42THEN280                              :rem 100
295 POKEX,61:POKEX+C,0:NEXTI                           :rem 7
300 FORI=1TOS1                                        :rem 88
305 X=INT(RND(1)*A)+B                                :rem 167
310 IFPEEK(X)=61THEN305                               :rem 92
315 IFPEEK(X)=42THEN305                               :rem 96
320 IFPEEK(X)=45THEN305                               :rem 95
325 POKEX,45:POKEX+C,2:NEXTI                           :rem 5
330 POKE7734,32:POKE7756,32:POKE8086,32              :rem 206
335 FORI=1TOS2                                        :rem 97
340 X=INT(RND(1)*A)+B                                :rem 166
345 IFPEEK(X)=61THEN340                               :rem 99
350 IFPEEK(X)=42THEN340                               :rem 94
355 IFPEEK(X)=45THEN340                              :rem 102
360 IFPEEK(X)=44THEN340                               :rem 97
365 POKEX,44:POKEX+C,5:NEXTI                          :rem 11
370 PRINT"{BLU}{HOME}{21 DOWN}{RIGHT}LOOT:$"         :rem 187
375 IFW=0THENTI$="000000"                            :rem 131
380 RETURN                                           :rem 123
385 FORT=1TO20:V=1:POKE36878,15                       :rem 87
390 FORL=1TO10                                        :rem 65
395 FORM=250TO240STEP-1                              :rem 124
400 POKE36876,M                                       :rem 74
405 POKE36879,26:NEXTM                                :rem 51
410 FORM=240TO250                                    :rem 214
415 POKE36876,M:POKE36879,41:NEXTM                    :rem 33
```

```
420 POKE36876,0:NEXTL                                   :rem 244
425 POKE36878,0                                          :rem 54
430 PRINT"{BLK}{CLR}{2 DOWN}{RIGHT}YOU GOT CAUGHT WITH":PRINT
    "{7 RIGHT}$";S                                       :rem 49
435 GOTO615                                             :rem 113
445 IFPEEK(Z-1)=45THENV=1                               :rem 243
450 IFPEEK(Z-1)=42THEN465                               :rem 199
455 POKEZ-1,43:POKEZ-1+C,4                                :rem 7
460 IFPEEK(Z+1)=45THENV=1                               :rem 238
465 IFPEEK(Z+1)=42THEN480                               :rem 200
470 POKEZ+1,43:POKEZ+1+C,4                                :rem 0
475 IFPEEK(Z+22)=45THENV=1                               :rem 39
480 IFPEEK(Z+22)=42THEN495                              :rem 254
485 POKEZ+22,43:POKEZ+22+C,4                            :rem 108
490 IFPEEK(Z-22)=45THENV=1                               :rem 38
495 IFPEEK(Z-22)=42THEN505                              :rem 254
500 POKEZ-22,43:POKEZ-22+C,4                            :rem 100
505 POKE36877,220                                       :rem 152
510 FORL=15TO0STEP-1                                    :rem 217
515 POKE36878,L                                          :rem 82
520 FORT=1TO25:NEXT                                     :rem 195
525 NEXTL                                                :rem 39
530 POKE36877,0:POKE36878,0                               :rem 7
535 IFV=1THEN385                                        :rem 191
540 IFPEEK(Z-1)=42THEN550                               :rem 194
545 POKEZ-1,32                                           :rem 22
550 IFPEEK(Z+1)=42THEN560                               :rem 194
555 POKEZ+1,32                                           :rem 21
560 IFPEEK(Z+22)=42THEN570                              :rem 247
565 POKEZ+22,32                                          :rem 73
570 IFPEEK(Z-22)=42THEN580                              :rem 251
575 POKEZ-22,32                                          :rem 76
580 POKEZ,59:RETURN                                     :rem 218
585 POKE36878,15                                        :rem 115
590 FORL=10TO1STEP-1                                    :rem 221
595 POKE36874,L*25:NEXTL                                :rem 172
600 POKE36874,0:POKE36878,0:RETURN                       :rem 28
605 POKE36879,30                                        :rem 106
610 PRINT"{CLR}{2 DOWN}{RIGHT}YOU MADE IT OUT WITH":PRINT"
    {7 RIGHT}{RED}$";S"{BLU}"                            :rem 38
615 IFV=1THENGOTO625                                    :rem 244
620 IFS>HSTHENHS=S                                      :rem 173
625 PRINT"{2 DOWN}{2 RIGHT}HIGH SCORE:$";HS             :rem 154
630 PRINT"{2 DOWN}{RIGHT}PRESS ANY KEY TO GO":PRINT"{7 RIGHT}
    AGAIN!"                                             :rem 151
635 GETA$:IFA$=""THEN635                                 :rem 95
640 GOTO25                                               :rem 58
645 DATA 7504 ,255,231,255,165,165,255,231,255,66,36,0,195,0,
    36,66,0                                             :rem 246
```

```
650 DATA 7632 ,0,0,0,16,0,0,16,0,24,24,60,126,189,60,36,102
                                                    :rem 4
655 DATA 7520 ,126,60,126,231,239,247,231,126,112,251,251,250
    ,116,40,112,248                                 :rem 105
660 DATA 7648 ,0,0,0,0,0,0,0,0,255,231,255,165,165,255,231,25
    5                                               :rem 163
665 DATA 0                                          :rem 235
```

V. Cortes

Diamond Maze

"Diamond Maze" is a maze game for the unexpanded VIC and requires a joystick. When the game begins, you find yourself in a dark maze trying to locate diamonds for your kingdom. You are provided with a small oil lamp to help you find your way inside the dark mazes. The oil in the lamp is limited, so you must try to get as many diamonds as you can before the light runs out. When it does, the quest is over.

The lamp will light the part of the maze that you are in as you move. What lies in any direction is unknown to you until you move in that direction. Once you make contact with a diamond, you are automatically transported to another and completely different maze where you continue your search for the next diamond. These diamonds glow brightly, so you will know their locations. By using the joystick to move in any available direction, you must find the quickest route. Each maze is numbered in the lower-right corner. This number represents the diamond number that you are going after. The amount of oil left in the lamp is also indicated on the right side of the screen. The indicator starts at the top and continues downward until it reaches the bottom of the screen, at which time the quest ends.

Making Mazes

In order to generate the maze quickly, this program uses a machine language subroutine. The maze is generated in the blink of an eye so that when you go from one maze to another, you will barely notice that the maze has changed. Any knowledge of the previous maze is of little use since each maze is different.

Lines 800–999 are the BASIC loader and DATA statements for the machine language maze-generating subroutine. This subroutine is loaded into the cassette buffer. Lines 60–640 initialize the screen to reverse spaces with a border. This is required by the machine language subroutine to determine the boundaries of the maze. These lines can be used to create your own mazes. The rest of the lines are used to implement the Diamond Maze logic.

Diamond Maze

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 SC=99                                                    :rem 118
20 GOSUB700:GOTO230                                        :rem 129
30 POKEDD,127:J=(PEEK(D1)AND60)OR(PEEK(D2)AND128):POKEDD,255
                                                            :rem 27
40 V=((JAND16)=0)-((JAND128)=0)+(((JAND4)=0)-((JAND8)=0))*22
                                                           :rem 121
50 B=A+V:IFPEEK(B)=90THENF=1:POKES1,220:GOTO80              :rem 56
```

```
60 IFTI>T2THENT2=TI+TK:POKET+C,6:T=T+22:POKET+C,1:IFT>TMTHEN2
   00                                                  :rem 47
70 IFPEEK(B)<>32THEN30                                 :rem 31
80 POKES1,240:POKEA,32:POKEB,81:POKES1,0              :rem 161
90 FORI=0TO8:POKEA+T(I)+C,6:NEXT:A=B                  :rem 231
100 FORI=0TO8:POKEA+T(I)+C,5:NEXT:IFF=0THEN30          :rem 34
110 F=0:M=M+1:GOSUB600:TK=360-M*30:GOTO100            :rem 188
200 PRINTCHR$(19);CHR$(158);" GAME OVER";:POKE36879,238
                                                       :rem 89
210 FORI=0TO8:POKEA+T(I)+C,1:NEXT                      :rem 18
220 IFPEEK(197)=64THEN220                             :rem 164
230 M=1:TI$="000000":TK=360                           :rem 155
240 T2=TI+TK:T=7701:GOSUB600:GOTO100                  :rem 174
600 PRINTCHR$(19);CHR$(31);CHR$(18);                    :rem 1
610 PRINT"********************.";                    :rem 145
620 FORI=1TO21:PRINT"-{19 SPACES}-.";:NEXT            :rem 227
630 PRINT"**********************";                   :rem 101
640 POKE8185,48+M:POKE8185+C,1                        :rem 185
650 SYS828:POKE36879,110:POKEA,81:POKET+C,1             :rem 59
660 X=INT(RND(1)*18)+2:Y=INT(RND(1)*19)+2               :rem 38
670 B=7680+Y*22+X:IFA=BTHEN660                        :rem 218
680 POKEB,90:POKEB+C,7:RETURN                           :rem 58
700 DD=37154:D2=DD-2:D1=37137:C=30720:A=7680+23       :rem 248
710 T(0)=-23:T(1)=-22:T(2)=-21:T(3)=-1:T(4)=0           :rem 37
720 T(5)=1:T(6)=21:T(7)=22:T(8)=23                       :rem 6
730 POKE36878,15:S1=36876:TM=8141                       :rem 90
800 B=828:REM BASIC LOADER                            :rem 240
810 READD:IFD<0THENRETURN                             :rem 129
820 POKEB,D:B=B+1:GOTO810                             :rem 222
828 DATA169,30,133,169,169,23,133,168,160,0           :rem 132
838 DATA145,168,173,36,145,101,167,109,40,145         :rem 232
848 DATA41,3,133,166,133,167,170,32,169,3               :rem 27
858 DATA201,32,176,32,169,160,209,170,208,26          :rem 178
868 DATA232,138,145,170,165,170,133,168,165,171         :rem 80
878 DATA133,169,165,166,9,4,170,32,169,3              :rem 250
887 IFSC<3THENPOKE7701+Y*22+C,3                         :rem 71
888 DATA169,32,145,170,208,202,230,166,165,166          :rem 30
898 DATA41,3,133,166,197,167,208,204,177,168          :rem 200
908 DATA133,166,169,32,145,168,165,166,201,5          :rem 186
918 DATA176,43,9,8,170,202,32,169,3,165               :rem 190
928 DATA170,133,168,165,171,133,169,208,159,24          :rem 34
938 DATA189,196,3,48,10,101,168,133,170,165           :rem 136
948 DATA169,105,0,208,8,101,168,133,170,165           :rem 127
958 DATA169,233,0,133,171,96,2,212,254,44               :rem 29
968 DATA255,22,1,234,254,44,2,212                     :rem 137
999 DATA -1                                             :rem 35
```

Edward
Tekeian

Meltdown

Your goal in "Meltdown," for the unexpanded VIC with a joystick, is to prevent a meltdown at the local power plant. You have 45 seconds to capture a capsule of nuclear fuel. Then you must weave your way through automatic defenses, reach the core of a nuclear reactor, and deliver the fuel capsule to prevent *meltdown* of the reactor. There is one hitch, however; once you drop off your fuel, you will only have delayed a meltdown for another 45 seconds. In order to stabilize the reactor and permanently stop meltdown, you must deliver 20 units of fuel to the core, and each one must be delivered within 45 seconds. Should you fail to deliver even a single capsule in the allotted time or lose all of your lives (you have three lives with a bonus life after ten capsules have been delivered), the reactor immediately goes meltdown and you lose the game.

You must pilot a Tritron Megatank around the screen to pick up fuel. The Tritron can move quickly in four directions, but it has one potentially dangerous or useful feature. When the joystick is in the neutral position, the Tritron will move to the right, heading for the reactor wall. It will continue until it hits one of the obstacles or until it hits the reactor wall. Although this is a hazard, it can also be used for speedy movement through clear zones. The last control is the fire button which, when pressed, will cause the Tritron to stop at a dead halt. By quickly pressing and releasing the button, you can cause a slow step-by-step movement, which is good for maneuvering through thickly cluttered areas of mines.

Obstacles

There are three obstacles which will destroy one of your tanks upon contact and also cost you precious time. The electric mines, which are cross-shaped characters spread out all over the screen, are your worst enemies, because you must carefully maneuver through them during all phases of the game. The other two obstacles are the red walls at the top and bottom of the screen and the left reactor wall. Any attempt to scroll off the left side of the screen at a spot other than the entrance to the reactor will cause instant destruction.

Entering the left side entrance will place you just above the core. This is the rear entrance and can be used if the regular entrance is blocked. It is also worth noting that touching the core without having captured fuel will destroy you.

A Few Words About Fuel

On each screen a red-and-white striped block will appear. This is the fuel you must collect. You need only touch it in order to capture it. Once you

have the fuel, a red diamond will appear in the upper-right side of the screen. This means that you can safely enter the reactor core. Once you capture a fuel capsule, it is yours until you reach the reactor. If you are destroyed then, you will be transported to the starting spot on the screen (minus a life) with a penalty of having time added to your score.

A Few Final Rules

As you progress through the screens, the level of difficulty increases so that the number of mines and warp discs will increase. Your starting position for each screen is the upper-left corner of the playfield facing the right reactor wall. You will also appear there after an explosion and at the start of each level.

Meltdown

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 POKE 36879,8                                          :rem 213
2 PRINT"{CLR}{2 DOWN}{5 RIGHT}{L}MN{K}ELTDOW{L}M{J}":POKE 368
  78,15                                                  :rem 77
3 FORG=118TO245:POKE36876,G:POKE36876,0:NEXT            :rem 172
4 F=62:S1=36875:E=30720:H=0:N=3:B=30                    :rem 226
5 PRINT"{CLR}"                                          :rem 153
6 POKE38401,5                                           :rem 198
8 FOR T=7680TO7768:POKE T,255:POKE T+E,0:NEXT           :rem 139
10 FOR S=7790TO8163:POKE S,96:NEXT                      :rem 244
11 PRINT"{HOME}FUEL UNITS:";H                           :rem 194
12 Q=7790:POKE36879,255                                  :rem 10
13 IFH=2THENB=60:IFH=4THENB=90:IF=H=8THENB=120:IFH=12THENB=16
   0:IFH=16THENB=200                                     :rem 70
15 FORG=1TOB:I=7834+INT(RND(1)*380)+1:POKE I,87:POKEI+E,4:NEX
   TG                                                   :rem 250
16 FORK=1TOB:Z=7790+INT(RND(1)*380)+1:POKEZ,65:POKEZ+E,15:NEX
   TK                                                   :rem 101
17 FOR U=7789TO8189 STEP 22:POKE U,95:POKE U+E,12:NEXT
                                                        :rem 107
18 FOR A=0TO21:POKE7768+A,111:POKE8164+A,111:POKE7768+A+E,2:P
   OKE8164+A+E,2:NEXT                                    :rem 73
20 D=7870+INT(RND(1)*294)+1:POKED,102:POKED+30720,234:rem 176
21 POKE7986,118:POKE7987,119:POKE8030,118:POKE8031,111:POKE80
   09,42:POKE38728,6                                    :rem 214
22 TI$="000000"                                         :rem 198
23 PRINT"{HOME}{3 DOWN}TRITRONS:";N                     :rem 196
24 Q=7790:POKEQ,F:POKEQ+E,15                            :rem 179
25 J=PEEK(37151):POKEQ,F:POKE Q+E,15                     :rem 77
28 PRINT"{HOME}{DOWN} TIME:"TI$                         :rem 137
29 IFJ=110THENQ=Q-1:POKEQ+1,32:POKEQ+E,15:F=60:POKE S1,150
                                                        :rem 180
30 IFJ=126ORJ=127THENQ=Q+1:POKEQ-1,32:POKEQ+E,15:F=62:POKE S1
   ,150                                                 :rem 119
```

```
31 IFJ=118THENQ=Q+22:POKEQ-22,32:POKEQ+E,15:POKE S1,130
                                                  :rem 246
33 IF J=122 THENQ=Q-22:POKEQ+22,32:POKEQ+E,15:POKES1,210
                                                  :rem 242
34 POKE S1,0                                      :rem 118
38 IF PEEK(Q)=102 THEN 200                         :rem 82
39 IF PEEK(Q)=42 AND PEEK(7744)=90 GOTO 400       :rem 247
41 IFPEEK(Q)=42 ANDPEEK(7744)<>90GOTO 100          :rem 42
42 IFPEEK(Q)=111THEN 100                           :rem 76
43 IFPEEK(Q)=65THEN100                             :rem 37
44 IFPEEK(Q)=87THEN 320                            :rem 46
45 IFTI$="000045"THEN 100                          :rem 35
47 IF PEEK(Q)=118THEN100:IF PEEK(Q)=119 THEN100:IFPEEK(Q)=111
    THEN100                                       :rem 160
49 IFPEEK(Q)=95THEN100                             :rem 46
50 GOTO 25                                          :rem 5
100 POKE36878,15                                   :rem 98
102 POKE36877,255                                 :rem 153
103 FORX=1TO255 STEP2:POKE36879,X:NEXT            :rem 101
105 POKE 36877,0                                   :rem 48
107 N=N-1:IFN=-1THEN125                            :rem 72
109 IFTI$=>"000035"THEN 125                       :rem 152
110 GOTO 23                                        :rem 48
125 PRINT"{6 DOWN}{6 RIGHT}{RED}QQQQQQQQ":PRINT"{6 DOWN}
    {3 RIGHT}{RED}{L}MN{K}ELTDOW{L}M{K}"          :rem 234
126 POKE36878,15:FORT=200TO255:POKE36877,T:NEXT    :rem 94
128 FOR K=1TO500:NEXT                             :rem 236
129 GOTO 125                                      :rem 109
140 :IFH=16THENB=+190                             :rem 189
160 POKE36878,15:PRINT"{CLR}{9 DOWN} MISSION ACOMPLISHED:
    {3 SPACES}MELTDOWN ABORTED"                   :rem 143
162 C=INT(RND(1)*255)+1                           :rem 221
164 S=INT(RND(1)*50)+175                           :rem 36
166 POKE36879,C                                    :rem 76
167 POKE36875,S                                    :rem 89
168 FORT=1TO100:NEXTT                              :rem 73
169 GOTO 162                                      :rem 114
200 POKE7744,90:POKE 7744+E,2                      :rem 57
201 POKE 36878,15:POKE36874,230:FORM=1TO50:NEXT   :rem 247
202 POKE36874,0                                    :rem 43
203 GOTO 25                                        :rem 53
320 Q=Q-7:POKEQ,81:POKEQ+7,32                     :rem 212
321 POKE36878,15:POKE36876,220                    :rem 158
322 FORJ=1TO50:NEXT                               :rem 183
323 POKE36876,0:GOTO 25                            :rem 11
400 POKE 36878,15                                 :rem 101
401 FORK=128 TO 255:POKE36874,K:NEXT               :rem 67
402 POKE 36874,0                                   :rem 45
430 H=H+1                                         :rem 192
433 IFH=20THEN 160                                :rem 214
434 IFH=10THENN=N+1                               :rem 116
435 POKE7744,32:GOTO10                              :rem 6
```

Ulrich
Merten

Rats! An Artificial Intelligence Simulation

Artificial intelligence, or AI, is the use of machines, especially digital computers, to perform tasks usually thought of as requiring intelligence when performed by humans. Interest in the subject was intense in the 1950s, when the potential of large computing machines was first being explored. Early successes were achieved with a program which proved theorems in symbolic logic, and with others which played creditable games of chess and checkers, but the immensity of the task of imitating human thought soon became apparent.

The databases demanded by these AI systems, and the complexity of the programs used to manipulate these databases, militate against their practical implementation in the present generation of microcomputers. It is, nevertheless, possible to make a simulation on the microcomputer which illustrates the methods of AI. In "Rats!," which is such a simulation for a VIC with 8K memory expansion, a rat solves a maze on the screen, in much the same way that an AI program seeks the solution to a problem.

Principal Features of AI Systems

The first step in using an AI system is to define the task to be performed as precisely as possible, specifying both the starting point and the goal.

Our rat starts from a specific position on the screen. Its target (perhaps a piece of cheese?) is hidden in a maze, and its goal may be only to reach the target, to reach the target as quickly as possible, or to find the shortest route to the target. Attaining one of these goals may require a somewhat different approach from that required to attain another, and in all cases, there may be more than one solution.

Getting from the starting position to the goal requires, in general, searching an information space, where each move adds something to our knowledge about that space. Practical AI systems tend to call upon very large, multidimensional databases. In our simulation, the information required is quite simple: Each of 506 locations on the screen is either closed or open, except two, which are marked as the starting position and the target, respectively. By determining which positions are open and how they are connected to other positions which are open, our rat can find its way to the target.

The organization of the information space can be of critical importance to the efficiency of an AI program. It is important to search the most significant space first and to be able to respond to new information without excessive backtracking through familiar territory. Our rat has ready access to information about its immediate environment, but no way, in most cases, of finding out about more distant points. Clearly, information about more distant parts of the maze, perhaps a road sign or two, would be useful, but the increase in the complexity of the information space necessary to provide this input is considerable.

Search through information spaces tends to become lengthy as the space expands, because each new point visited can lead to several possible next moves. If the resulting combinatorial explosion is to be kept in hand, it is essential that whatever constraints exist on the solution be applied at the earliest possible point. The most obvious constraint on the solution of our problem is that the target is somewhere on the screen. Our rat is constrained to investigate the 506 screen positions and those positions only. If the right constraints were not in the program, the rat would investigate memory locations that have no relation to the screen positions and therefore sure to be unrewarding.

Search Strategies

A central feature of every AI program is its search strategy. A highly inefficient strategy, but one which will eventually explore all points in the available information space, is a random walk from one point in that space to any other which is accessible from the first, and so on. Among the more efficient alternatives are the *breadth-first* and *depth-first* strategies. Our rat can illustrate both of these.

With the breadth-first strategy the rat will travel the same distance down each available path, in rotation, going one step further each time it traverses a given path. In this fashion it is sure to find the shortest route to the target; but it may go through a great deal of wasted motion in the process.

For a depth-first search the rat will explore a given path until the path comes to an end or until the rat reaches the goal; then the rat moves to the next available path if it was unsuccessful. In many situations this strategy is more efficient than the breadth-first strategy. But there is no guarantee that the path found will be the shortest one possible. A great deal of time can be wasted in cases where a short path exists, but along with much longer successful paths and/or dead ends.

Many more sophisticated strategies are available. The *branch-and-bound* strategy, for instance, combines some of the strengths of the depth-first and the breadth-first approaches. The path of the rat following a simple branch-and-bound requires that it proceed down an available route until a branch

point is reached. Then the rat reviews the available data and continues along a path emanating from the previously discovered intersection which is closest to the starting point, and from which an unexplored path is still available. This branch-and-bound approach is also sure to find the shortest route to the goal, and frequently with much less wasted motion than is inherent in the breadth-first search.

As we have already indicated, searches can be more efficient when information about more distant parts of the information space is available. In particular, the branch-and-bound strategy can be improved if some estimate can be made of the remaining distance to the target, and if the choice of path can be based on an estimate of the total path length rather than on knowledge of the distance already traveled. This modified strategy will find the shortest path provided that the estimate is always less than or equal to the remaining distance. It will further reduce the amount of search required if the estimates are accurate than if they are inaccurate. (An estimate of zero remaining distance always meets the requirement that it be less than the actual remaining distance, but provides no improvement over the original strategy.)

We can illustrate the impact of an estimate by attributing a sense of smell to our rat and assume that it can estimate the distance from the cheese by noting the intensity of its aroma. Since all path lengths through the maze must be longer than the straight-line distance from rat to cheese, this will be an appropriate estimate to use in the branch-and-bound strategy, and will cause our rat to quickly abandon paths which are moving it away from the target.

A second feature of every AI program is a control mechanism for guiding the chosen search strategy and, in some cases, selecting among available strategies. A common control method is the *production system* which consists of a series of IF-THEN statements which test for specified conditions and take action accordingly.

Our rat uses a production system at each step, first to determine whether the target is at hand, and then to find a direction in which it can proceed without encountering a wall. IF the rat is by the target, THEN it eats the cheese. IF it finds an open space, THEN it moves into the space. IF it finds itself in a blind alley, THEN it turns around and goes back.

The efficiency of the search may be strongly influenced by the order in which the productions are triggered. It is generally desirable to trigger productions first which have the potential for eliminating large portions of the search space as unfruitful, and then testing subtler questions. Triggering productions in a systematic order can in some cases lead to a result different from a random triggering.

This last effect is easily demonstrated by our rat. If we program the rat to select from its options at each intersection in a random fashion, after a

sufficiently long series of runs through the maze, it will have examined every possible path. If, on the other hand, we have the rat look straight ahead first, then left, then right, say, at every intersection, it will still find its way to the target, but on repeated runs it will always find the same route, which may or may not be the shortest one available.

Leaving a Trail

In many AI programs, it is not only important to reach the goal, but also to leave an *audit trail* which documents the path taken. Such a trail is essential if the user wants to ask why the program reached the indicated conclusion.

Our rat needs an audit trail for two reasons: During any given run, a trail is helpful to prevent repetitious exploration of blind alleys. And if several runs are made to find the shortest path, each path must be remembered for comparison with previous paths, and the shortest must be remembered so that the rat can find it again.

Since keeping track of where it has been puts a significant strain on the rat's (and the VIC's) memory, we have explored several techniques to keep track of things. In one case, the depth-first search, we equip the rat with a mental map of the entire search space so that it can mark the points visited. This is a very adequate approach for avoiding duplication and remembering the path actually chosen, but it requires a secondary counting of steps to determine the length of the path. In other cases, the rat counts steps and remembers directions from the starting point, but it is not provided any information about its absolute position in the search space.

All AI programs have an ability to learn, in a certain sense, but they vary widely in the extent to which they must be *spoon-fed* their new knowledge. In most cases, new knowledge is incorporated into the program by a human programmer who enlarges the information space, or improves the search strategy or the control. But some programs, particularly game programs, are self-modifying; they avoid past mistakes the next time around.

Our rat exhibits two kinds of learning ability and could be equipped with yet a third. First, it learns with human intervention when the program is changed. (We find the rat to be a slow learner when we do the programming!) Second, it learns from experience in any given maze and retains what was learned in this audit trail. It can find and remember the shortest route to the target. And it could surely learn to select an optimum search strategy if we were to write the appropriate program.

The Program

Rats! requires at least 11K of RAM (8K or more memory expansion). The program is written in BASIC. It provides for storing mazes on tape or disk, but the use of external storage is not essential, since the user can construct

appropriate mazes as a part of running the program. Tape users should change OPEN1,8,2,"MAZE" in line 800 to OPEN1,1,0,"MAZE" and also change OPEN1,8,2,"MAZE,S,W" in line 900 to OPEN1,1,1,"MAZE".

Integer variables are used extensively to minimize the demand on RAM. (An integer variable consumes two bytes of VIC memory; a real number, five bytes.) The size of the R% array is an important constraint on the mazes which can be explored by the program. If more than ten possible paths exist or if the shortest path is more than 100 steps long, the maze will not be accepted.

Using the Program

The meaning and utility of various search strategies and auditing techniques can be explored by using the maze construction routines to fashion mazes having various desired characteristics, and then letting our rat run through those mazes using the various search routines.

Many more configurations are possible. The branch-and-bound method with a remaining-distance estimate will be found to be especially powerful in mazes in which paths tend to lead in a consistent direction, but this technique is less useful in mazes where early steps down the correct path appear to be in the wrong direction. The breadth-first routine will be impressive only in cases where a short path to the target coexists with much longer paths, none of which need be fully explored using this strategy.

For a given search, we may set as a goal finding the target as quickly as possible on the first run or finding the shortest path to the target. In most cases, the depth-first strategy will be found quite effective at finding the target quickly on the first run of the maze, but it is a characteristic of the memory device used that the best route may never be found except after repeated searches from scratch. In depth-first searches, our rat will prefer its first successful route unless we wipe its memory clean and let it use the random-selection routine another time.

The score sheet accessible at the end of each run makes it possible to compare the rat's performance in a given maze using a variety of search strategies, and before and after learning in each strategy. The record is cleared each time we return to the main menu, and such returns are necessary only to alter the maze.

References

Alexander, T. "Teaching Computers the Art of Reason." *Fortune* (May 17, 1982): 82.

Barr, A., and E. A. Feigenbaum, eds. *The Handbook of Artificial Intelligence.* Los Altos, CA: William Kaufmann, 1981.

Lindsay, R. K., B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg. *Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project.* New York: McGraw-Hill, 1980.

McCorduck, P. *Machines Who Think.* San Francisco: W. H. Freeman, 1972.

Newell, A. *Production Systems: Models of Control Structures in Visual Information Processing.* Edited by W. G. Chase. New York: Academic Press, 1972.

Nilsson, N. J. *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga, 1980.

Raphael, B. *The Thinking Computer.* San Francisco: W. H. Freeman, 1976.

Shortliffe, E. H. *Computer-Based Medical Consultations: MYCIN.* New York: American Elsevier, 1976.

Winograd, J. *Understanding Natural Language.* New York: Academic Press, 1972.

Winston, P. H. *Artificial Intelligence.* Reading, MA: Addison-Wesley, 1977.

Rats!

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
30 DIMB%(506),T%(506),R%(100,10),D%(4),C%(4)          :rem 102
31 DEFFNA(X)=INT((X-1)/22)-INT((X2-1)/22):DEFFNB(X)=ABS(X-X2-
   22*FNA(X))+ABS(FNA(X))                              :rem 253
50 PRINT"{CLR}{5 SPACES}{RVS}RATS{2 SPACES}MENU{OFF}":PRINT:P
   RINT:PRINT"A{3 SPACES}CONSTRUCT MAZE":PRINT:PRINT  :rem 54
53 PRINT"B{3 SPACES}GET MAZE FROM FILE":PRINT:PRINT"C
   {3 SPACES}MODIFY MAZE":PRINT:PRINT                 :rem 116
57 PRINT"D{3 SPACES}RUN MAZE":PRINT:PRINT:PRINT"E{3 SPACES}FI
   LE MAZE":PRINT:PRINT                                :rem 92
60 PRINT"F{3 SPACES}END SESSION"                      :rem 120
68 V=0:FORK=1TO10:S%(K)=0:NEXT                         :rem 67
70 GETZ$:IFZ$=""THEN70                                 :rem 35
71 IFZ$="A"THEN100                                    :rem 247
72 IFZ$="B"THEN800                                      :rem 0
73 IFZ$="C"THEN150                                      :rem 0
74 IFZ$="D"THEN200                                    :rem 254
75 IFZ$="E"THEN900                                      :rem 7
77 IFZ$="F"THENEND                                     :rem 72
100 PRINT"{CLR}{3 SPACES}{RVS}MAZE CONSTRUCTION{OFF}":PRINT:P
    RINT"PRESS K TO OPEN SPACE":PRINT                  :rem 35
102 PRINT"PRESS D TO CLOSE SPACE":PRINT:PRINT"{3 SPACES}TO MO
    VE AND OPEN"                                      :rem 107
104 PRINT"{4 SPACES}ADJACENT SPACE":PRINT"{5 SPACES}PRESS:"
                                                      :rem 222
106 PRINT"{11 SPACES}I{2 SPACES}UP":PRINT:PRINT"{3 SPACES}LEF
    T{2 SPACES}J Z L{2 SPACES}RIGHT":PRINT             :rem 8
```

```
108 PRINT"{5 SPACES}DOWN{2 SPACES}M":PRINT:PRINT:PRINT"PRESS
    {SPACE}1 FOR START":PRINT                          :rem 130
112 PRINT"PRESS 2 FOR TARGET":PRINT:PRINT"{2 SPACES}{RVS}PRES
    S Z TO CONTINUE{OFF}"                               :rem 55
120 GETZ$:IFZ$=""THEN120                               :rem 123
148 FORK=1TO506:B%(K)=160:NEXT                           :rem 5
150 FORK=1TO506:POKE37887+K,6:POKE4095+K,B%(K):NEXT     :rem 31
151 PRINT"{HOME}{21 DOWN}":PRINT"PRESS Q TO RETURN";    :rem 135
152 FORV=1TO10:S%(V)=0:NEXT:V=0                         :rem 131
154 X=243:B=32                                          :rem 230
155 IFX<23THENX=X+22:GOTO157                            :rem 222
156 IFX>484THENX=X-22                                    :rem 14
157 POKE4095+X,B:B%(X)=B                                 :rem 88
159 GETZ$:IFZ$=""THEN159                                :rem 147
160 IFZ$="D"THENB=160                                   :rem 174
165 IFZ$="K"THENB=32                                    :rem 136
176 IFZ$="L"THENX=X+1:B=32                               :rem 14
177 IFZ$="J"THENX=X-1:B=32                               :rem 15
178 IFZ$="M"THENX=X+22:B=32                              :rem 68
179 IFZ$="I"THENX=X-22:B=32                              :rem 67
180 IFZ$="1"THENB=90:X1=X                               :rem 199
181 IFZ$="2"THENB=83:X2=X                               :rem 204
183 IFZ$="Q"THEN50                                       :rem 15
184 GOTO155                                             :rem 113
200 PRINT"{CLR}{5 SPACES}{RVS}SEARCH MENU":PRINT:PRINT:PRINT"
     A{3 SPACES}RANDOM":PRINT:PRINT                      :rem 28
202 PRINT" B{3 SPACES}BREADTH FIRST":PRINT:PRINT:PRINT" C
    {3 SPACES}BRANCH AND BOUND":PRINT:PRINT             :rem 140
204 PRINT" D{3 SPACES}B AND B PLUS":PRINT:PRINT:PRINT" E
    {3 SPACES}DEPTH FIRST":PRINT:PRINT                  :rem 175
206 PRINT" Q{3 SPACES}MAIN MENU"                         :rem 20
210 GETZ$:IFZ$=""THEN210                                :rem 123
211 IFZ$="A"THENJ=1:GOTO250                              :rem 84
212 IFZ$="B"THENJ=2:GOTO300                              :rem 83
213 IFZ$="C"THENJ=3:GOTO350                              :rem 91
214 IFZ$="D"THENJ=4:GOTO400                              :rem 90
215 IFZ$="E"THENJ=5:GOTO450                              :rem 98
216 IFZ$="Q"THEN50                                       :rem 12
217 GOTO210                                             :rem 102
250 WW=0:F=32                                           :rem 212
251 GOSUB700:GOTO255                                    :rem 190
253 GOSUB500                                            :rem 175
255 IFL=0THEN260                                        :rem 171
256 GOSUB731:T%(W)=T%(W-1)+D:GOTO290                     :rem 76
260 GOSUB510                                            :rem 174
261 IFN=0THENGOSUB720:GOTO265                            :rem 59
262 P=INT(RND(1)*N)+1                                   :rem 157
263 D=D%(P):X=X+D                                       :rem 188
265 IFW=1THENT%(1)=D:GOTO267                            :rem 174
266 T%(W)=T%(W-1)+D                                     :rem 234
267 IFW<3THEN280                                        :rem 189
```

```
268 FORK=1TOW-1                                    :rem 152
269 IFT%(K)=T%(W)THEN273                           :rem 110
271 NEXT                                           :rem 217
272 GOTO280                                        :rem 110
273 W=K                                            :rem 123
280 GOSUB730:GOTO253                               :rem 193
290 IFWW=0THENWW=W                                 :rem 182
292 IFW>WWTHENGOTO550                              :rem 113
293 WW=W:T=1                                       :rem 220
294 FORK=1TOWW                                     :rem 144
295 R%(K-100*(P-1),P)=T%(K)                        :rem 104
296 IFK-100*(P-1)=100THENP=P+1                     :rem 152
297 NEXT                                           :rem 225
298 WP=P                                           :rem 215
299 GOTO550                                        :rem 119
300 F=46:WW=0:GG=1                                  :rem 11
301 GOSUB700                                       :rem 171
303 WW=WW+1:IFWW<101THEN305                        :rem 151
304 GOSUB982:GOTO317                               :rem 200
305 FORG=1TOGG                                     :rem 101
307 IFE%(G)=1THEN340                                :rem 95
310 GOSUB740:GOSUB500                              :rem 254
316 IFL=0THEN320                                   :rem 166
317 R%(WW,G)=D:GOSUB731:GS=G:GOTO550               :rem 191
320 GOSUB510                                       :rem 171
321 IFN=0THENE%(G)=1:GOSUB720:GOSUB750:GOTO340      :rem 50
323 R%(WW,G)=D%(1)                                 :rem 169
324 IFN=1THEN335                                   :rem 174
325 IF(GG+N-1)<11THEN327                            :rem 72
326 G=10:GOSUB980:GOTO317                          :rem 233
327 FORK=2TON                                       :rem 46
328 FORW=1TOWW-1:R%(W,GG+K-1)=R%(W,G):NEXT          :rem 39
329 R%(WW,GG+K-1)=D%(K)                            :rem 228
330 NEXT                                           :rem 213
331 GG=GG+N-1                                      :rem 199
335 GOSUB750                                       :rem 183
340 NEXTG                                           :rem 29
345 GOTO303                                        :rem 107
350 F=46:GG=1:GS=1                                 :rem 253
352 GOSUB700                                       :rem 177
354 WW=500                                          :rem 28
356 FORG=1TOGG                                     :rem 107
358 IFE%(G)=1THEN362                               :rem 105
360 IFW%(G)<WWTHENGS=G:WW=W%(G)                    :rem 172
362 NEXT                                           :rem 218
364 G=GS:GOSUB740:X=X+R%(WW,GS):GOSUB731           :rem 243
366 WW=WW+1:IFWW<101THEN368                        :rem 169
367 GOSUB982:GOTO372                               :rem 210
368 W%(GS)=WW:GOSUB500                              :rem 66
370 IFL=0THEN374                                   :rem 175
372 GOSUB731:R%(WW,GS)=D:GOTO550                   :rem 187
```

```
374 GOSUB510                                                    :rem 180
376 IFN=0THENE%(GS)=1:GOSUB720:GOSUB750:GOTO354                 :rem 148
380 R%(WW,GS)=D%(1)                                             :rem 255
382 IFN=1THENX=X+D%(1):GOSUB731:GOTO366                         :rem 129
384 FORK=2TON                                                    :rem 49
385 GG=GG+1:IFGG<11THEN387                                       :rem 27
386 GOSUB980:GOTO372                                            :rem 209
387 W%(GG)=W%(GS)                                               :rem 161
388 FORW=1TOWW-1:R%(W,GG)=R%(W,GS):NEXT                         :rem 172
390 R%(WW,GG)=D%(K)                                              :rem 14
392 NEXT                                                        :rem 221
396 GOSUB720:GOSUB750:GOTO354                                    :rem 32
400 F=46:GG=1:GS=1:WW=0:T%(1)=X1                                 :rem 73
402 GOSUB700                                                    :rem 173
404 VV=500                                                       :rem 22
406 FORG=1TOGG                                                  :rem 103
408 IFW%(G)+FNB(T%(G))+E%(G)*500<VVTHENGS=G:VV=W%(G)+FNB(T%(G
    ))                                                           :rem 93
410 NEXT                                                        :rem 212
412 WW=W%(GS)                                                   :rem 233
414 G=GS:GOSUB740:X=X+R%(WW,GS):GOSUB731                        :rem 239
416 WW=WW+1:IFWW<101THEN418                                     :rem 161
417 GOSUB982:GOTO422                                            :rem 202
418 W%(GS)=WW:GOSUB500                                           :rem 62
420 IFL=0THEN424                                                :rem 167
422 GOSUB731:R%(WW,GS)=D:GOTO550                                :rem 183
424 GOSUB510                                                    :rem 176
426 IFN=0THENE%(GS)=1:GOSUB720:GOSUB750:GOTO404                 :rem 140
428 R%(WW,GS)=D%(1):T%(GS)=X                                     :rem 53
430 IFN=1THENX=X+D%(1):GOSUB731:GOTO416                         :rem 119
432 FORK=2TON                                                    :rem 43
433 GG=GG+1:IFGG<11THEN435                                       :rem 15
434 GOSUB980:GOTO422                                            :rem 199
435 W%(GG)=W%(GS):T%(GG)=T%(GS)                                 :rem 206
436 FORW=1TOWW-1:R%(W,GG)=R%(W,GS):NEXT                         :rem 166
438 R%(WW,GG)=D%(K)                                              :rem 17
440 NEXT                                                        :rem 215
442 GOSUB720:GOSUB750                                             :rem 9
444 GOTO404                                                     :rem 109
450 F=32:FORK=1TO506:T%(K)=6:NEXT                               :rem 212
452 GOSUB700                                                    :rem 178
454 GOSUB500:IFL=0THEN460                                       :rem 253
458 GOSUB731:GOTO550                                            :rem 201
460 N=0:GOSUB520:IFN=0THEN466                                   :rem 249
462 P=INT(RND(1)*N)+1                                           :rem 159
464 X=X+D%(P):POKE37887+X,C%(P):T%(X)=C%(P):GOSUB731:GOTO454
                                                                :rem 249
466 GOSUB530:IFN=0THEN472                                         :rem 8
468 P=INT(RND(1)*N)+1                                           :rem 165
470 X=X+D%(P):GOSUB731:GOTO454                                   :rem 34
472 GOSUB540:IFN=0THEN476                                        :rem 10
```

```
474 POKE37887+X,7:T%(X)=7:X=X+D:GOSUB731:GOTO454      :rem 144
476 GOSUB720:GOSUB731:GOTO454                          :rem 31
500 IFB%(X+1)=83THENX=X+1:D=1:L=1:RETURN              :rem 170
501 IFB%(X+22)=83THENX=X+22:D=22:L=1:RETURN            :rem 68
502 IFB%(X-22)=83THENX=X-22:D=-22:L=1:RETURN          :rem 118
503 IFB%(X-1)=83THENX=X-1:D=-1:L=1:RETURN             :rem 222
505 RETURN                                            :rem 122
510 N=0                                                :rem 81
512 IFB%(X+1)=32THENN=N+1:D%(N)=1                      :rem 73
513 IFB%(X-22)=32THENN=N+1:D%(N)=-22                  :rem 223
514 IFB%(X+22)=32THENN=N+1:D%(N)=22                   :rem 177
515 IFB%(X-1)=32THENN=N+1:D%(N)=-1                    :rem 123
519 RETURN                                            :rem 127
520 IFB%(X+1)=32ANDT%(X+1)=6THENN=N+1:D%(N)=1:C%(N)=0:rem 186
522 IFB%(X-22)=32ANDT%(X-22)=6THENN=N+1:D%(N)=-22:C%(N)=2
                                                      :rem 136
524 IFB%(X+22)=32ANDT%(X+22)=6THENN=N+1:D%(N)=22:C%(N)=3
                                                       :rem 90
526 IFB%(X-1)=32ANDT%(X-1)=6THENN=N+1:D%(N)=-1:C%(N)=4
                                                      :rem 245
528 RETURN                                            :rem 127
530 IFT%(X+1)=0THENN=N+1:D%(N)=1                       :rem 38
532 IFT%(X-22)=2THENN=N+1:D%(N)=-22                   :rem 191
534 IFT%(X+22)=3THENN=N+1:D%(N)=22                    :rem 147
536 IFT%(X-1)=4THENN=N+1:D%(N)=-1                      :rem 95
538 RETURN                                            :rem 128
540 IFB%(X+1)=32ANDT%(X+1)<5THEND=1:N=1               :rem 207
542 IFB%(X-22)=32ANDT%(X-22)<5THEND=-22:N=1           :rem 155
544 IFB%(X+22)=32ANDT%(X+22)<5THEND=22:N=1            :rem 108
546 IFB%(X-1)=32ANDT%(X-1)<5THEND=-1:N=1                :rem 6
548 RETURN                                            :rem 129
550 FORK=1TO7:POKE38384+K,6:NEXT                      :rem 216
551 POKE4593,16:POKE4594,18:POKE4595,5:POKE4596,19:POKE4597,1
    9:POKE4598,32:POKE4599,26                           :rem 2
552 L=0:GETZ$:IFZ$=""THEN552                          :rem 128
555 POKE4095+U,F:POKE4095+Z,F:POKE4095+Y,F:POKE4095+X,F
                                                       :rem 37
556 POKE4095+X1,90:POKE4095+X2,83                     :rem 113
557 B%(U)=F:B%(Z)=F:B%(Y)=F:B%(X)=F:B%(X1)=90:B%(X2)=83
                                                      :rem 224
560 PRINT"{CLR}";                                      :rem 58
561 ONJGOTO562,563,564,565,566                        :rem 135
562 PRINT"{4 SPACES}{RVS}RANDOM SEARCH{OFF}":GOTO567 :rem 158
563 PRINT"{4 SPACES}{RVS}BREADTH FIRST{OFF}":GOTO567 :rem 170
564 PRINT"{3 SPACES}{RVS}BRANCH AND BOUND{OFF}":GOTO567
                                                       :rem 34
565 PRINT"{3 SPACES}{RVS}BRANCH AND BOUND{OFF}":PRINT"
    {4 SPACES}{RVS}PLUS ESTIMATE{OFF}":GOTO567        :rem 114
566 PRINT"{5 SPACES}{RVS}DEPTH FIRST{OFF}"             :rem 19
567 PRINT:PRINT                                       :rem 246
```

```
571 PRINT"A{3 SPACES}RERUN - NO MEMORY":PRINT:PRINT"B
    {3 SPACES}RERUN WITH MEMORY":PRINT                    :rem 90
573 PRINT"C{3 SPACES}CHECK SCORES":PRINT:PRINT"D{3 SPACES}GO
    {SPACE}TO SEARCH MENU":PRINT                         :rem 225
574 IFJ=5THENPRINT"E{3 SPACES}COLOR MEMORY":PRINT         :rem 79
575 PRINT"Q{3 SPACES}GO TO MAIN MENU"                     :rem 86
586 GETZ$:IFZ$=""THEN586                                 :rem 155
587 IFZ$="C"THENGOSUB950:GOTO560                         :rem 208
589 ONJGOTO600,620,640,660,680                           :rem 120
600 IFZ$="B"THEN610                                       :rem 44
601 FORP=1TO10:FORK=1TO100:R%(K,P)=0:NEXT:NEXT           :rem 118
602 FORK=1TO506:T%(K)=0:NEXT                             :rem 171
603 IFZ$="A"THEN251                                       :rem 47
604 IFZ$="D"THEN200                                       :rem 45
605 IFZ$="Q"THEN50                                        :rem 14
609 GOTO560                                              :rem 115
610 GOSUB700                                             :rem 174
612 FORP=1TO10                                            :rem 66
613 FORK=1TO100                                          :rem 110
614 GOSUB731                                             :rem 182
615 X=X1+R%(K,P)                                         :rem 116
616 IFK+100*(P-1)=WWTHENGOTO619                           :rem 79
617 NEXT                                                 :rem 221
618 NEXT                                                 :rem 222
619 GOSUB731:S%(V)=S%(V)-1:GOTO551                       :rem 220
620 IFZ$="B"THEN627                                       :rem 54
621 GOSUB760                                             :rem 182
622 FORK=1TOGG:E%(K)=0:FORW=1TOWW:R%(W,K)=0:NEXT:NEXT:rem 126
623 IFZ$="A"THEN300                                       :rem 44
624 IFZ$="D"THEN200                                       :rem 47
625 IFZ$="Q"THEN50                                        :rem 16
626 GOTO560                                              :rem 114
627 GOSUB700                                             :rem 182
628 G=GS:GOSUB740:X=X+R%(WW,G):GOSUB731:GOTO551          :rem 177
640 IFZ$="B"THEN654                                       :rem 56
642 GOSUB760                                             :rem 185
644 FORG=1TOGG:FORW=1TOW%(G):R%(W,G)=0:NEXT:E%(G)=0:W%(G)=0:N
    EXT                                                  :rem 151
646 IFZ$="A"THEN350                                       :rem 54
648 IFZ$="D"THEN200                                       :rem 53
650 IFZ$="Q"THEN50                                        :rem 14
652 GOTO560                                              :rem 113
654 GOSUB700                                             :rem 182
656 G=GS:GOSUB740:X=X+R%(WW,G):GOSUB731:GOTO551          :rem 178
660 IFZ$="B"THEN674                                       :rem 60
662 GOSUB760                                             :rem 187
664 FORG=1TOGG:FORW=1TOW%(G):R%(W,G)=0:NEXT:E%(G)=0:W%(G)=0:T
    %(G)=0:NEXT                                           :rem 81
666 IFZ$="A"THEN400                                       :rem 52
668 IFZ$="D"THEN200                                       :rem 55
670 IFZ$="Q"THEN50                                        :rem 16
```

```
672 GOTO560                                                  :rem 115
674 GOSUB700                                                 :rem 184
676 G=GS:GOSUB740:X=X+R%(WW,G):GOSUB731:GOTO551              :rem 180
680 IFZ$="B"THENFORK=1TO506:POKE4095+K,B%(K):POKE37887+K,T%(K
    ):NEXT:GOSUB702:GOTO454                                  :rem 104
682 IFZ$="E"THEN695                                           :rem 70
684 FORX=1TO506:POKE37887+X,6:T%(X)=6:NEXT                    :rem 52
686 IFZ$="A"THEN450                                           :rem 59
688 IFZ$="D"THEN200                                           :rem 57
690 IFZ$="Q"THEN50                                            :rem 18
692 GOTO560                                                  :rem 117
695 FORK=1TO506:IFB%(K)=32ANDT%(K)=6THENPOKE4095+K,32:POKE378
    87+K,6:GOTO697                                           :rem 237
696 POKE4095+K,160:POKE37887+K,T%(K):POKE4095+X1,90:POKE4095+
    X2,83                                                     :rem 27
697 NEXT:GOTO550                                             :rem 242
700 FORK=1TO506:POKE4095+K,B%(K):POKE37887+K,6:NEXT           :rem 32
702 X=X1:U=X:Y=X:Z=X                                          :rem 44
705 IFV=10THENFORN=1TO9:S%(N)=S%(N+1):NEXT:GOTO709            :rem 90
707 V=V+1                                                    :rem 227
709 W=1:S%(V)=0:RETURN                                        :rem 69
720 POKE4095+U,F:B%(U)=F:POKE4095+X,M:B%(X)=M                 :rem 97
725 D=Y-X:U=Z:Z=X:X=Y:RETURN                                 :rem 142
730 W=W+1                                                    :rem 225
731 S%(V)=S%(V)+1                                            :rem 114
732 M=67:IFABS(Y-Z)=22THENM=93                                :rem 41
737 POKE4095+U,F:POKE4095+Z,M:B%(U)=F:B%(Z)=M                :rem 109
738 POKE4095+Y,81:POKE4095+X,90:B%(Y)=81:B%(X)=90:U=Z:Z=Y:Y=X
    :RETURN                                                  :rem 130
740 X=X1:Y=X:Z=X:U=X:IFWW=1THENRETURN                         :rem 34
741 FORK=1TOWW-1:X=X+R%(K,G):GOSUB731:NEXT                   :rem 145
742 RETURN                                                   :rem 125
750 FORK=WW-1TO1STEP-1:X=X-R%(K,G):GOSUB731:NEXT              :rem 45
751 POKE4095+U,F:POKE4095+Z,F:B%(U)=F:B%(Z)=F:RETURN         :rem 117
760 FORK=23TO483                                             :rem 179
762 IFB%(K)=46THENB%(K)=32                                   :rem 172
763 NEXT                                                     :rem 223
765 RETURN                                                   :rem 130
800 PRINT"{CLR}GET MAZE FROM FILE":PRINT:OPEN1,8,2,"MAZE"
                                                              :rem 32
810 FORK=1TO506                                              :rem 119
811 INPUT#1,B%(K)                                            :rem 173
812 IFB%(K)=90THENX1=K                                        :rem 19
813 IFB%(K)=83THENX2=K                                        :rem 23
814 NEXT                                                     :rem 220
815 CLOSE1:GOTO50                                             :rem 29
900 PRINT"{CLR}FILE MAZE":PRINT:OPEN1,8,2,"MAZE,S,W"          :rem 15
911 FORX=1TO506:PRINT#1,B%(X):NEXT                            :rem 86
915 PRINT#1:CLOSE1:GOTO50                                     :rem 57
950 PRINT"{CLR}{3 SPACES}{RVS}LATEST RAT SCORES{OFF}":PRINT:P
    RINT"TRIAL","SCORE":PRINT                                 :rem 42
```

```
955 FORK=1TO10                                          :rem 71
956 IFS%(K)=0THEN960                                   :rem 130
957 PRINTK,S%(K)                                       :rem 189
958 NEXT                                               :rem 229
960 PRINT:PRINT:PRINT" PRESS Z TO CONTINUE"            :rem 237
965 GETZ$:IFZ$=""THEN965                               :rem 157
970 RETURN                                             :rem 128
980 GG=10:PRINT"TOO MANY BRANCHES":RETURN               :rem 95
982 WW=100:PRINT"PATH TOO LONG":RETURN                 :rem 147
999 END                                                :rem 130
```

Chapter Eight

Shoot-em-ups

Paul
Austin

Space Corridor

Get ready for the ultimate video experience. "Space Corridor" is not just a game, but an adventure where your skills as a pilot are tested to their fullest. All you need is an unexpanded VIC and a joystick to test your skills.

To manipulate your ship, move joystick left and right. The fire button activates the laser. Be wary of enemy bombs and ships. Watch your fuel consumption (monitored by the gauge near the top of screen). After you shoot five of the enemy's ships, your fuel is restored and you set up base, but your base is constantly blasted by other enemy ships.

Typing It In

When typing in this program, be sure to save a copy before you run it since an error in one of the DATA statements could cause the program to crash. Program 2 disables the RUN/STOP key.

If you are using a disk, be sure to save Program 2 with the filename SPACE.PRG. Tape users should save Program 2 on the same tape immediately after Program 1. Tape users should also delete lines 63995 and 63997 of Program 1 and replace line 63999 with

63999 POKE198,1:POKE631,131

Good luck and happy laser blasting!

Program 1. Space Corridor

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
15 POKE56,27:POKE55,240:POKE51,240:POKE52,240:CLR     :rem 104
20 FORT=7416TO7583:READC:POKET,C:NEXT:FORT=7152TO7413:READC:P
   OKET,C:NEXT                                        :rem 149
999 DATA231,231,231,231,231,231,231,231               :rem 169
1000 DATA0,0,0,0,0,0,0,0,231,231,231,195,219,24,66,126:rem 87
1010 DATA57,57,57,57,131,199,239,239,255,221,221,221,235,247,
     247,255                                           :rem 95
1020 DATA255,219,219,219,231,231,255,255,255,255,235,235,247,
     247,247,247                                       :rem 37
1030 DATA255,255,255,235,247,255,255,255,255,255,255,247,255,
     255,255,255                                       :rem 50
1050 DATA255,255,255,255,255,255,255,255               :rem 244
1060 DATA0,2,10,11,47,47,43,42,170,186,190,234,254,255,255,25
     5                                                 :rem 22
1070 DATA0,128,160,160,168,168,168,168,170,170,170,170,170,17
     0,170,170                                         :rem 171
1080 DATA255,255,254,238,234,250,186,190,170,170,170,170,170,
     170,170,170                                       :rem 16
1085 DATA42,42,42,42,10,10,2,0,190,255,255,190,190,186,186,18
     6                                                 :rem 27
```

```
1086 DATA168,168,168,168,160,160,128,0,126,189,219,255,255,21
     9,189,126                                          :rem 216
1087 DATA231,231,231,231,231,231,231,231                :rem 206
1090 REMINTERRUPT DATA                                  :rem 149
1110 DATA172,60,3,192,255,240,39,169,127,141,34,145,169,40,15
     3,100,31,173,17                                    :rem 200
1120 DATA145,41,16,208,1,136,173,32,145,41,128,208,1,200,169,
     33,153,100,31                                       :rem 76
1130 DATA140,60,3,169,255,141,34,145,76,191,234          :rem 57
1140 REM ***ML DATA***                                   :rem 89
1150 DATA169,151,133,253,169,206,133,252,162,0,169,151,133,25
     5,169,228                                          :rem 180
1160 DATA133,254,162,0,160,0,177,252,145,254,200,192,22,208,2
     47,56                                              :rem 217
1165 DATA165,254,233,22,133,254,165,252,233,22,133,252,232,22
     4,10,208,227                                        :rem 50
1170 DATA173,61,3,160,0,153,8,151,200,192,22,208,248:REM SHIP
      ML                                                 :rem 12
1175 DATA169,31,133,255,169,0,133,254,160,0,177,254,201,50,20
     8,10,165,2,201,31,208,2                             :rem 66
1176 DATA169,40,145,254,201,51,208,4,169,40,145,254,201,31,20
     8,33,169,40,145,254                                :rem 150
1177 DATA192,230,176,25,152,24,105,22,168,177,254,201,33,208,
     5,169,1,133,0                                      :rem 105
1178 DATA96,169,31,145,254,152,56,233,22,168,136,208,192,169,
     0,141,11,144                                        :rem 78
1180 DATA169,3,133,1,173,17,145,41,32,208,55,169,1,205,62,3,2
     40,47,169                                          :rem 166
1190 DATA1,141,62,3,174,60,3,138,56,233,22,170,188,100,31,192
     ,40,240,13                                         :rem 191
1200 DATA132,2,169,50,157,100,31,169,255,141,11,144,96,169,51
     ,157,100,31,198,1                                   :rem 39
1210 DATA208,222,169,200,141,11,144,96,169,0,141,62,3,96
                                                        :rem 236
63995 POKE 36879,15                                     :rem 226
63997 PRINT "{CLR}{YEL}PLEASE WAIT....{BLK}":PRINT "{2 DOWN}L
      OAD";CHR$(34);"SPACE.PRG";CHR$(34);",8"           :rem 101
63999 PRINT "{5 DOWN}RUN":PRINT"{HOME}":POKE198,2:POKE631,13:
      POKE 632,13                                       :rem 221
```

Program 2. Space.prg

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 POKE809,242:PRINTCHR$(144)CHR$(147)SPC(3)"SPACE CORRIDOR!"
   :GOSUB2000                                           :rem 176
30 FORT=7584TO7679:POKET,255:NEXT:POKE.,.                :rem 34
40 POKE37159,255:PRINTCHR$(18)" HIT ANY KEY TO BEGIN ";
                                                         :rem 35
50 WAIT197,255,64:POKE828,94:PRINTCHR$(147)             :rem 242
55 C=2:FORT=7922TO8185STEP22                            :rem 187
```

```
57 FORJ=1TOC:POKET+(11-J/2),40:POKET+11+J/2,40:NEXT:C=C+2:NEX
   T                                                    :rem 240
60 PRINTCHR$(17)CHR$(17)CHR$(17)CHR$(17)CHR$(17)CHR$(17)CHR$(
   17)"{2 DOWN}"                                        :rem 111
65 POKE36867,46:POKE36878,143:POKE36879,8:POKE36869,255
                                                        :rem 154
70 PRINTCHR$(5);:FORT=1TO10:PRINTSPC(10-T)CHR$(18)"{D}";SPC(2
   *T);"{F}":NEXT                                       :rem 253
80 PRINTCHR$(18)CHR$(28)CHR$(19)CHR$(17)"FUEL:{OFF}456789:;<=
   >?":POKE646,11                                       :rem 103
85 PRINTCHR$(17):POKE37166,64:POKE788,240:POKE789,27:POKE3716
   6,192                                                :rem 106
90 PRINTCHR$(17)SPC(14")")*+":PRINTSPC(14)",-.":PRINTSPC(14)"/
   01"                                                  :rem 115
100 NK=.:SC=.:FL=7588:A=256:V=36874:L1=8120:L2=8141  :rem 101
105 Y=7932:CH=40:GOSUB470                               :rem 188
106 IFPEEK(Y)=50THENSYS7201:SC=SC+10:POKEY,40:NK=NK+1:POKE368
    75,220:GOTO105                                       :rem 84
107 IFNK=5THENGOSUB3000:SC=SC+50                        :rem 215
110 POKEY,40::Y=Y+INT(RND(1)*3)+21:CH=CH-.5:IFPEEK(Y)=255THEN
    Y=Y+23                                                :rem 1
112 IFPEEK(Y)=33THENPOKE828,255:POKEY,50:AN=250:GOSUB1500:ER=
    2:GOTO1540                                           :rem 74
113 IFPEEK(L2)=33THENPOKE828,255:POKEL1,50:AN=250:GOSUB1500:E
    R=2:GOTO1540                                        :rem 148
114 IFPEEK(L1)=33THENPOKE828,255:POKEL1,50:AN=250:GOSUB1500:E
    R=2:GOTO1540                                        :rem 148
120 POKEY,INT(CH)                                        :rem 14
121 IFCH=38THENPOKEY+22,31                               :rem 44
129 POKE829,INT(RND(1)*7)+1                             :rem 249
130 IFCT=2THENA=A/2:POKEFL,PEEK(FL)-A:POKEFL-1,PEEK(FL-1)-A:C
    T=1                                                 :rem 173
131 IFA=1THENFL=FL+8:A=256:IFFL>7680THENAN=.:GOSUB1500:ER=1:G
    OTO1540                                             :rem 174
132 CT=CT+1:SYS7201                                      :rem 96
133 IFPEEK(.)=1THENA=PEEK(828):POKE828,255:POKE8036+A,50:AN=1
    00:GOSUB1500:ER=2:GOTO1540                          :rem 155
140 POKEV,300+NOT(3*CH)                                  :rem 46
150 IFY>8200THEN105                                      :rem 75
160 GOTO106                                             :rem 103
470 PRINTCHR$(19)CHR$(18)CHR$(30)"SCORE";SC,"HIGH";HI:RETURN
                                                         :rem 91
900 DATA231,231,231,195,219,24,66,126                    :rem 79
1500 REM SOUND ROUTINE                                   :rem 89
1510 POKE36876,200:POKE36877,200                        :rem 248
1520 FORT=15TO0STEP-1                                    :rem 19
1530 POKE36878,T+128:FORTJ=TTOAN:NEXT:NEXT:POKE36876,.:POKE36
     877,.                                              :rem 101
1535 RETURN                                             :rem 174
1540 REM CRASH                                           :rem 31
```

```
1550 PRINTCHR$(5)CHR$(147)CHR$(18);:IFER=1THENPRINT" YOU ARE
     {SPACE}OUT OF FUEL"                                  :rem 31
1551 IFER=2THENPRINT"{2 SPACES}YOU HAVE CRASHED"         :rem 124
1552 PRINTCHR$(17)CHR$(18)" YOUR FORCE HAS BEEN{7 SPACES}DEST
     ROYED"CHR$(17)                                       :rem 52
1553 POKE36867,34                                         :rem 158
1560 PRINTCHR$(18)CHR$(17)"YOU WILL HAVE TO START";       :rem 215
1570 PRINTSPC(8)"AGAIN."CHR$(17)CHR$(17)                  :rem 15
1580 POKE37166,197:IFSC>HITHENHI=SC                       :rem 146
1590 FORT=7432TO7439:READC:POKET,C:NEXT:RESTORE           :rem 45
1600 GOTO30                                               :rem 99
2000 POKE808,199:POKE36869,242:PRINTCHR$(17)CHR$(8)" YOU ARE
     {SPACE}EXPLORING THE";                               :rem 21
2010 PRINT"PLANET 'FREON' LOOKING";                       :rem 238
2020 PRINT"FOR LIFE ON THIS "                             :rem 113
2030 PRINT"PLANET."                                       :rem 136
2040 PRINT" ON THE PLANET YOU "                           :rem 86
2050 PRINT"FIND A NATION OF"                              :rem 88
2060 PRINT"DRUIDS WHO HOPE TO USE";                       :rem 73
2070 PRINT"THEIR 'SLOW BOMBS' AND";                       :rem 42
2080 PRINT"THEIR SPACECRAFT ON A"                         :rem 209
2090 PRINT"RECKLESS RUN TO WIPE"                          :rem 197
2100 PRINT"OUT YOUR SHIP BEFORE"                          :rem 194
2110 PRINT"YOU CONQUER THEIR "                            :rem 43
2120 PRINT"PLANET."                                       :rem 136
2130 PRINT"YOUR{SHIFT-SPACE}MISSION:TO TAKE"              :rem 170
2140 PRINT"OVER THE PLANET BY"                            :rem 20
2150 PRINT"DESTROYING THEIR AIR"                          :rem 249
2160 PRINT"FORCE OF 5 SHIPS"                              :rem 90
2170 PRINT"BEFORE THEY HIT YOU OR";                       :rem 70
2180 PRINT"YOU RUN OUT OF FUEL."                          :rem 117
2190 POKE36879,25:RETURN                                  :rem 185
3000 NK=.:PRINTCHR$(18)CHR$(19)"{3 SPACES}YOUR FORCE HAS
     {2 SPACES}"                                          :rem 227
3010 PRINTCHR$(18)"DEFEATED THE AIR FORCE";               :rem 233
3020 PRINT" YOUR BASE IS NOW SET"                         :rem 124
3030 PRINTCHR$(18)"UP. YOU MUST NOW FIGHT";               :rem 12
3040 PRINT"GORKS FROM{2 SPACES}'PLANET Z'":POKE37166,64
                                                          :rem 196
3050 FORT=1TO25:POKE36876,230:FORJ=1TO40:NEXT:POKE36876,.:FOR
     J=1TO40                                              :rem 189
3060 NEXT:NEXT:PRINTCHR$(19);:FORT=1TO5:PRINT"{22 SPACES}";
                                                          :rem 64
3070 NEXT:FORT=7432TO7439:POKET,.:NEXT:FL=7588:A=256      :rem 207
3080 FORT=7584TO7679:POKET,255:NEXT:GOSUB470:RETURN       :rem 8
```

Jerry
Fennell

Base Defense

"Base Defense" is an entertaining game for the unexpanded VIC with a joystick.

It's an easy game to play. The object is quite simply to destroy as many meteors as you can with five bases. The closer the meteors get, the more points they're worth because of the greater risk. Point distribution is as follows: 50 points for close range, 25 for midscreen, and 10 for faraway shots.

It's that simple, and it even leaves 2K for you to add your own extras.

Base Defense

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}":POKE36879,14:POKE37139,0                :rem 156
15 GOSUB2500:GOSUB2000                                   :rem 41
17 PRINT"{CLR}{22 DOWN}"                                 :rem 66
20 R=7690:V=36878:S1=36877:S2=36876:G=5:DD=37154        :rem 174
30 FORT=1TO3                                            :rem 226
40 GOSUB200                                             :rem 118
50 P=INT(RND(1)*22)                                      :rem 34
60 POKE8142+P,1                                          :rem 60
65 POKE38862+P,INT(RND(1)*7)+1                          :rem 171
70 NEXT T                                               :rem 250
72 IFPEEK(R+22)=1THENGOSUB1000                           :rem 45
75 PRINT"{DOWN}":GOSUB80:GOTO30                          :rem 70
80 POKE R,0:RETURN                                       :rem 95
200 P=PEEK(37137)                                       :rem 154
205 IF(P AND 32)=0THENGOTO400                           :rem 104
210 IF(P AND 16)=0THEN DR=-1:GOTO250                    :rem 212
215 POKEDD,127                                           :rem 21
220 M=PEEK(37152)                                       :rem 150
225 IF(M AND 128)=0 THEN DR=1:POKEDD,255:GOTO250        :rem 151
230 DR=0:POKEDD,255                                      :rem 81
250 POKE R,32                                           :rem 169
255 R=R+DR                                               :rem 62
260 IF R<=7680THENR=7680                                 :rem 90
265 IF R>=7701THENR=7701                                 :rem 85
270 POKE R,0                                            :rem 118
275 RETURN                                              :rem 126
400 POKEV,15:FOR C=1TO20                                :rem 130
405 POKER+22*C,46                                       :rem 172
410 POKE S2,230-C                                       :rem 122
415 IF PEEK((R+22*C)+22)=1 THEN POKE (R+22*C)+22,2:GOSUB500:S
    C=SC+AD:GOTO450                                     :rem 211
420 POKE R+22*C,32                                      :rem 164
425 POKE S2,0:NEXT C                                    :rem 103
450 FORW=1TO25:NEXTW:POKE(R+22*C)+22,32                  :rem 71
455 POKE R+22*C,32                                      :rem 172
460 RETURN                                              :rem 122
```

```
500 POKEV,15:POKES1,215:POKES2,200                    :rem 9
510 IFC<5THENAD=50                                   :rem 47
515 IFC>=5ANDC<=15THENAD=25                         :rem 106
517 IFC>15THENAD=10                                 :rem 101
520 POKEV,0:POKES1,0:POKES2,0                         :rem 11
530 RETURN                                          :rem 120
1000 POKEV,15:POKES1,215:POKES2,150                   :rem 57
1002 POKE 36879,239                                 :rem 205
1005 FOR X=1 TO 5                                     :rem 75
1006 POKE R+X,62:POKE R-X,60                          :rem 49
1007 NEXT X                                           :rem 95
1010 FORE=15TO0STEP-1                               :rem 254
1012 POKEV,E                                        :rem 186
1014 FORW=1TO100:NEXTW                              :rem 118
1016 NEXTE                                            :rem 76
1020 POKEV,0:POKES1,0:POKES2,0                        :rem 55
1030 G=G-1:IFG=0THEN GOTO 1500                      :rem 152
1035 POKE 36879,14                                  :rem 154
1040 RETURN                                         :rem 165
1500 PRINT"{CLR}":POKE36879,27:POKE37154,255:POKE36869,240
                                                    :rem 169
1510 PRINT"{BLU}YOUR SCORE WAS ";SC                   :rem 62
1520 INPUT"WANT TO PLAY AGAIN";A$                   :rem 175
1530 IF LEFT$(A$,1)="N" THEN END                    :rem 143
1540 PRINT"{CLR}"                                     :rem 46
1550 SC=0:CLR:POKE36869,255:POKE36879,14:GOTO17       :rem 16
2000 PRINT"{RVS}GENERATING GRAPHICS"                :rem 218
2005 FORQ=7168TO7679:POKEQ,PEEK(Q+25600):NEXTQ        :rem 49
2010 FOR X=7168 TO 7191:READ A:POKE X,A:NEXT X        :rem 32
2020 DATA 66,66,66,102,102,126,60,24,60,34,123,141,129,81,34,
     28                                               :rem 16
2030 DATA 36,157,66,36,24,129,36,66,0,0,0           :rem 217
2040 POKE 36869,255                                 :rem 205
2050 RETURN                                         :rem 167
2500 PRINT:PRINTTAB(5);"{WHT}BASE DEFENSE"            :rem 17
2521 FORX=0TO21:POKE7680+X,42:POKE8164+X,42:NEXTX   :rem 241
2522 FORX=0TO22:POKE7680+22*X,42:POKE7701+22*X,42:NEXT X
                                                      :rem 11
2530 FORW=1TO2000:NEXTW                             :rem 171
2540 PRINT"{CLR}":RETURN                              :rem 73
```

Chris Hill

Bomber Squadron

"Bomber Squadron" is a game in two parts for the unexpanded VIC and a joystick. It's easy to play. All the instructions appear on the screen.

On this fateful night you will receive your first wartime mission. When the papers finally arrive, you feel a sharp pain in the pit of your stomach. You have been selected for the Bomber Squadron.

Suddenly, you find yourself deep over enemy lines. All you can hear is the dull roar of the plane's engines as you gaze into the radar. Then you see them, the enemy installations with a little more than you bargained for—jamming devices. Of course, since you are such a shrewd, level-headed person, you view these as only minor obstacles. You make a mental note to avoid them while positioning your sights, for your sights are at the only part of the plane where bombs can get out or jamming signals in. It's all up to you now.

Just as you bomb the first installation, the pilot turns around and warns you, "The targeting computer is having a problem. If you run off the top or bottom of the radar, it will not correct it." You realize what this means. If you exit the radar's screen, the system will crash. Havoc will reign, and you will surely go down. But enough worrying for now. The enemy awaits.

Custom Characters

Program 1 is merely a simple program to enter custom characters into the VIC. If you are unfamiliar with custom characters, you may wish to consult the *VIC-20 Programmer's Reference Guide* or *COMPUTE!'s First Book of VIC*. They both give good explanations of custom characters. With minimal changes, Program 1 can be customized for your own games. All you need to do is alter the DATA statements, following these simple rules:

1. The first number of each DATA statement contains the POKE value of the character to be replaced. For example, line 100 begins with a zero. This means we are replacing the @ character.
2. The next eight numbers define exactly what the new character will look like.
3. Continue repeating this process until you have defined all the characters you wish to define.
4. After you've defined all the characters with DATA statements, add another DATA statement (or add this to the end of your last DATA statement) and place a −1 there (as in line 190). This will exit the loop.

Typing the Program

Program 1 will automatically run Program 2. Disk users should save Program 2 with the filename BOMBER.PRG. Tape users should save Program 2 on the same tape immediately following Program 1, and should delete lines 60, 70, and 80 of Program 1 and replace line 50 with the following:

50 POKE198,1:POKE631,131

Some of the longer lines of the program need to be entered with keyword abbreviations. Use a question mark (?) for PRINT and N SHIFT-E for NEXT. For a complete list of abbreviations, see you computer manual.

Program 1. Bomber Squadron

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}{2 SPACES}CHARACTER GENERATOR":PRINT"{5 SPACES}
   BY CHRIS HILL"                                        :rem 65
20 POKE52,28:POKE56,28:CLR:FORI=7168TO7679:POKEI,PEEK(I+25600
   ):NEXT                                                :rem 78
30 CB=7168:READA:IFA=-1THEN50                            :rem 203
40 FORN=0TO7:READB:POKE(CB+A*8+N),B:NEXT:GOTO30          :rem 187
50 POKE198,2:POKE631,13:POKE 632,13                      :rem 187
55 POKE 36879,8                                           :rem 14
60 PRINT "{CLR}{YEL}GET READY.......{BLK}":PRINT "{2 DOWN}LOA
   D";CHR$(34);"BOMBER.PRG";CHR$(34);",8"                :rem 244
70 PRINT "{5 DOWN}RUN"                                   :rem 130
80 PRINT "{HOME}"                                         :rem 76
100 DATA0,0,0,120,124,124,120,0,0                         :rem 79
110 DATA1,255,129,249,254,254,249,129,255                 :rem 33
120 DATA2,255,129,129,129,129,129,129,255                 :rem 31
125 DATA3,24,60,60,24,90,36,24,36                        :rem 126
130 DATA49,60,126,126,126,126,60,60,126                  :rem 170
140 DATA50,24,60,60,60,60,60,24,60                       :rem 164
150 DATA51,0,16,56,56,56,16,56,0                          :rem 80
160 DATA52,0,0,16,56,56,16,56,0                           :rem 23
170 DATA53,0,0,16,16,16,16,0,0                           :rem 214
180 DATA54,0,0,0,16,16,0,0,0                             :rem 106
190 DATA55,0,0,0,16,0,0,0,0,-1                           :rem 191
```

Program 2. Bomber.prg

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 POKE36879,8:POKE36869,255                               :rem 22
5 LOC=7955:SC=0:COL=38664+11                              :rem 84
10 PRINT"{CLR}{BLU}{RVS}*** BOMBER SQUADRON***"           :rem 22
20 FORT=1TO2000:NEXT                                     :rem 233
30 PRINT"{CLR}{4 SPACES}{RVS}INSTRUCTIONS?"              :rem 205
```

```
40 GETA$:IFA$=""THEN40                                    :rem 235
50 IFA$="N"THEN99                                         :rem 201
60 IFA$<>"Y"THEN40                                          :rem 4
70 PRINT"{CLR}{RVS}{2 SPACES}YOU ARE NOW A WORLD WAR II BOMBE
   R PILOT!{2 SPACES}AS YOU FLY OVER THE"                 :rem 228
71 PRINT"{RVS}ENEMY LINES, YOU MUST ATTEMPT TO BOMB ALL
   {3 SPACES}THE INSTALLATIONS THAT";                     :rem 163
72 PRINT"{RVS}PASS UNDERNEATH YOU.{2 SPACES}THE {OFF}B{RVS} I
   S YOUR CROSS{3 SPACES}HAIRS. USE THE"                   :rem 75
73 PRINT"{RVS}JOYSTICK TO AIM YOUR{2 SPACES}BOMBS, AND THE FI
   RE{3 SPACES}BUTTON TO BOMB."                           :rem 204
74 PRINT"{RVS}AVOID THE {OFF}C{RVS} OR A{6 SPACES}SERIOUS MAL
   FUNCTION{3 SPACES}MAY OCCUR."                          :rem 102
99 PRINT"{RVS}PRESS FIRE TO BEGIN."                        :rem 62
100 GOSUB1000:IFFB=0THEN100                                :rem 80
109 PRINT"{CLR}{WHT}"                                       :rem 3
120 GOSUB210                                              :rem 166
130 GOSUB250                                              :rem 171
139 TI$="000000"                                          :rem 255
140 GOSUB370                                              :rem 175
150 GOSUB1000                                             :rem 215
151 IFFBTHENGOSUB410:GOTO191                                :rem 0
152 POKELOC,32                                             :rem 54
160 IFJ2THENLOC=LOC-1                                      :rem 40
170 IFJ0THENLOC=LOC+1                                      :rem 37
180 IFJ3THENLOC=LOC-22                                     :rem 94
190 IFJ1THENLOC=LOC+22                                     :rem 91
191 IFLOC=JMTHEN9000                                      :rem 212
192 IFLOC>8163ORLOC<7702THENGOSUB2000                      :rem 21
195 IFPEEK(LOC)<>0THENPOKELOC,2:GOTO201                   :rem 204
196 POKELOC,1                                              :rem 10
201 IFTI$=>"000030"THENGOSUB5000                           :rem 57
209 GOTO140                                               :rem 105
210 REM ENGINE & SIGHTS                                    :rem 37
220 POKE36878,15:POKE36877,241                            :rem 160
230 POKELOC,32:XX=PEEK(LOC):IFXX=0THENPOKELOC,1:J=1:RETURN
                                                           :rem 57
235 POKELOC,2:J=0:RETURN{19 SPACES}                        :rem 16
250 REM ESTABLISH BLDGS.                                  :rem 180
260 BB=7681:A=BB+INT(RND(0)*20):B=BB+INT(RND(0)*20):C=BB+INT(
    RND(0)*20)                                            :rem 238
270 D=BB+INT(RND(0)*20):JM=BB+INT(RND(0)*20)              :rem 224
300 Z=INT(RND(1)*22):X=INT(RND(1)*6)+1                    :rem 135
320 IFX=1THENA=A+Z                                         :rem 93
330 IFX=2THENB=B+Z                                         :rem 97
340 IFX=3THENC=C+Z                                        :rem 101
350 IFX=4THEND=D+Z                                        :rem 105
355 IFX=5THENJM=JM+Z                                       :rem 21
360 POKEA,0:POKEB,0:POKEC,0:POKED,0:POKEJM,3:RETURN       :rem 246
370 REM MOVE BLDGS.                                        :rem 79
```

```
380 POKEA,32:POKEB,32:POKEC,32:POKED,32:POKEJM,32:A=A+22:B=B+
    22:C=C+22:D=D+22:JM=JM+22                                  :rem 68
390 GG=8163:IFA>GGTHEN250                                     :rem 213
391 IFB>GGTHEN250                                              :rem 0
392 IFC>GGTHEN250                                              :rem 2
393 IFD>GGTHEN250                                              :rem 4
394 IFJM>GGTHEN250                                             :rem 88
400 POKEA,0:POKEB,0:POKEC,0:POKED,0:POKEJM,3:RETURN           :rem 241
410 REM BOMB DROP                                             :rem 206
411 IFPEEK(LOC)=0ORPEEK(LOC)=1THENJ=1:GOTO420                  :rem 57
415 J=0                                                        :rem 81
420 BO=48:VO=36876:NO=200                                     :rem 160
421 FORT=1TO7:BO=BO+1:NO=NO-1:POKELOC,BO:POKEVO,NO:FORH=1TO20
    0:NEXT:NEXT                                               :rem 247
425 POKEVO,0                                                  :rem 203
430 IFJ<>1THENRETURN                                           :rem 42
440 REM BOMB HITS!                                            :rem 245
450 VO=36877:HH=16:POKEVO,0:POKEVO,135:HH=HH-1:FORT=1TO15:POK
    E36878,HH:FORQW=1TO100                                    :rem 182
451 NEXTQW:NEXT:POKEVO,0:POKEVO,241:POKE36878,15               :rem 64
452 SC=SC+1                                                    :rem 96
453 RETURN                                                    :rem 124
1000 DD=37154:P1=37151:P2=37152                               :rem 126
1001 POKEDD,127:P=PEEK(P2)AND128                              :rem 108
1002 J0=-(P=0)                                                :rem 181
1003 POKEDD,255:P=PEEK(P1)                                      :rem 1
1004 J1=-((PAND8)=0)                                           :rem 20
1005 J2=-((PAND16)=0)                                          :rem 69
1006 J3=-((PAND4)=0)                                           :rem 20
1007 FB=-((PAND32)=0)                                          :rem 81
1008 RETURN                                                   :rem 169
2000 POKE36877,0                                               :rem 92
2001 POKE36879,110:PRINT"{CLR}{RVS}CONTROL TOWER STATUS
     {2 SPACES}REPORT...":FORT=1TO1500:NEXTT                  :rem 134
2002 PRINT"{RVS}{2 SPACES}YOU OVERLOADED THE{2 SPACES}FLIGHT
     {SPACE}COMPUTER!"                                        :rem 184
2003 PRINT"{RVS}YOU DID DESTROY ";SC;"{5 SPACES}INSTALLATIONS
     ."                                                         :rem 3
2004 PRINT"{2 DOWN}{RVS}CONTROL TOWER SIGNING OFF...":FORT=1T
     O9999:NEXT:CLR:GOTO1                                     :rem 147
5000 IFTI$=>"000200"THENGOTO6000                               :rem 36
5010 IFTI$=>"000130"THENPOKE36879,93:RETURN                    :rem 26
5020 IFTI$=>"000100"THENPOKE36879,93:RETURN                    :rem 24
5030 IFTI$=>"000030"THENPOKE36879,42:RETURN                    :rem 21
6000 REM SUCCESSFUL MISSION                                   :rem 204
6100 POKE36877,0:POKE36879,110:PRINT"{CLR}{RVS}{WHT}CONTROL T
     OWER STATUS{2 SPACES}REPORT..."                           :rem 74
6150 FORT=1TO1500:NEXT                                         :rem 87
6200 PRINT"{RVS}SUCCESSFUL MISSION{4 SPACES}COMPLETED."
                                                              :rem 152
```

```
6201 PRINT"{RVS}"SC" INSTALLATIONS{8 SPACES}DESTROYED."
                                                        :rem 76
6202 PRINT"{RVS}PRESS FIRE TO REPLAY"                  :rem 208
6203 GOSUB1000:IFFB=0THEN6203                          :rem 196
6204 GOTO1                                              :rem 54
9000 REM JAM                                           :rem 133
9010 POKE36877,0:POKE36879,110:PRINT"{CLR}{RVS}{WHT}CONTROL T
     OWER STATUS{2 SPACES}REPORT..."                    :rem 77
9015 FORT=1TO1500:NEXT                                  :rem 90
9020 PRINT"{RVS}COMPUTER SYSTEMS{6 SPACES}JAMMED. SYSTEM
     {8 SPACES}MALFUNCTION. PLANE"                     :rem 244
9030 PRINT"{RVS}DOWNED BEHIND ENEMY{3 SPACES}LINES."   :rem 65
9040 PRINT"{RVS}NEW PILOT FOR MISSION NEEDED. PRESS FIRE."
                                                      :rem 223
9050 GOSUB1000:IFFB=0THEN9050                         :rem 202
9060 GOTO1                                             :rem 57
```

Marcus
Warren
Hobbs

Ex-Blast

You're an Ex-Blaster, one of the last of your race. Only a few Blasters are left. The Orions, who destroyed most of your people with a surprise bombing, are planning another attack. This will be the last one, for you know that if this attack is successful, it will be the end of your kind. The Orions have built hundreds of cities in outer space. These cities are guarded by Orion Warriors. Fortunately, the Orions have not built laser cannons, but they do have powerful bombs. The bombers need fuel to complete their dastardly mission. It's your job to destroy those fuel pods with your laser cannon, without being damaged three times by running into such objects as the deadly Electro-walls, Spikes, Orion Scouts, Orion Warriors, stars, and fuel pods.

A Difficult Task

"Ex-Blast" is a tough game, but someone has got to save the Ex-Blast people. To play, you'll need a joystick, an unexpanded VIC, and a good sense of timing. As the aliens attack, use the joystick to move out of their way and to align yourself with the attacker when you wish to fire. To shoot your laser, simply press the fire button.

The joystick will allow movement in all four directions. If you find the stick is not functioning properly, press the RUN/STOP–RESTORE key combination and rerun the program. The stick should respond correctly.

Ex-Blast

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

Some lines of this program require keywords to be abbreviated so that they will not exceed the four-screen-line limit. See Appendix B.

```
Ø POKE36881,128                                          :rem 48
2 L=1:POKE37154,255                                      :rem 33
5 FORT=7424TO7436:POKET,Ø:NEXT:POKE36878,15             :rem 146
1Ø CLR:POKE52,28:POKE56,28:POKE36879,8:FORT=7168TO7255:READA:
   POKET,A:NEXT:FORA=ØTO65:READB:POKE828+A,B            :rem 13Ø
11 NEXT                                                 :rem 161
2Ø PRINT"{CLR}{12 DOWN}{6 RIGHT}{GRN}EX-BLAST!"         :rem 191
3Ø FORT=128TOØSTEP-.5:POKE36881,T:NEXT                  :rem 129
4Ø FORT=ØTO5Ø:POKE36881,T:POKE36879,T:NEXT:FORT=5ØTOØSTEP-1:P
   OKE36879,T:POKE36881,T:NEXT                          :rem 1ØØ
5Ø GETA$:IFA$=""THEN4Ø                                  :rem 236
51 A$(Ø)="JJJJJJJJJJJJJJJJJJJJ":Z=Ø:J=1Ø:Q=Ø            :rem 25Ø
58 A$(6)="@IJJJJJJJJJJJJJJJJJI@"                        :rem 214
6Ø A$(1)="@IJJJIIIIIIIIIIIJJJI@":POKE36881,24:POKE36869,255
                                                          :rem 5
61 A$(2)="@IIIJJJIIIIIIIIIIIII@":POKE36879,8            :rem 156
62 A$(3)="@IIIIIIIIIIIIIIJJJIII@"                       :rem 192
64 A$(4)="@II@I@I@I@I@I@I@I@II@":PRINT"{CYN}"            :rem 34
```

```
65 A$(5)="@IICIAIDIAIAIDIAICII@"                          :rem 140
66 SD$="{HOME}{DOWN}{LEFT}{INST}":PRINT"{CLR}":MP=8141:F=100:
   T=10:G=7:H=8:SD=218:P3=158:P1=37157:PRINT"{CLR}"   :rem 193
67 DD=37154:C=30720                                       :rem 236
71 POKE36877,252:FORPQ=1TO100:POKEMP,G                     :rem 72
72 SYS828:MP=MP+PEEK(1)-PEEK(2):POKEMP,G:POKEMP+22,H:TE=TE-.2
   :IFTE<0THENTE=0                                         :rem 29
73 IFMP>8185ORMP<7680THENGOSUB501                          :rem 62
74 IFRND(1)<.3THEN76                                      :rem 153
75 GOTO81                                                  :rem 14
76 PP=INT(RND(1)*4):ONPPGOTO77,78,79,80                    :rem 51
77 E$="F":GOTO81                                          :rem 122
78 E$="D":GOTO81                                          :rem 121
79 E$="C":GOTO81                                          :rem 121
80 E$="A"                                                 :rem 147
81 POKEDD,255:IFPEEK(37151)=94THENGOSUB3000                :rem 68
83 POKEMP,J:POKEMP+22,J:POKEMP+1,J:POKEMP-1,J:POKEMP+44,J
                                                          :rem 121
84 PRINT"{HOME}{DOWN}";A$(0):PRINTTAB(INT(RND(1)*23));E$:PRIN
   TSD$:POKESD,P3:POKE646,INT(RND(1)*6+1)                  :rem 10
86 O=PEEK(MP):IFO=1ORO=3ORO=4ORO=6THENGOSUB501           :rem 252
87 PRINTSD$:POKESD,P3:IFPEEK(MP)<>JANDPEEK(MP)<>32THENGOSUB50
   1                                                       :rem 16
88 NEXT                                                   :rem 175
179 POKE36877,252:FORPQ=1TO110                              :rem 9
180 TE=TE-.2:SYS828:MP=MP+PEEK(1)-PEEK(2):IFMP>8185ORMP<7680T
    HENGOSUB501                                           :rem 214
181 POKEMP,G:POKEMP+22,H                                  :rem 226
182 POKEDD,255:IFPEEK(37151)=94THENGOSUB3000              :rem 118
190 Z=Z+1:IFZ>6THENPOKE646,INT(RND(1)*6+1):PRINT"{HOME}{DOWN}
    ";A$(INT(RND(1)*6)):Z=0:GOTO210                        :rem 47
200 PRINT"{HOME}{DOWN}"A$(6)                              :rem 115
203 TE=TE-.3:SYS828:MP=MP+PEEK(1)-PEEK(2):IFMP>8185ORMP<7680T
    HENGOSUB501                                           :rem 211
209 IFRND(1)<.2THENPOKE 7702+INT(RND(1)*22),2              :rem 19
210 POKEMP,J:POKEMP+22,J:POKEMP+23,J:PRINTSD$:POKESD,P3:IFPEE
    K(MP)=JORPEEK(MP)=32THENNEXT                           :rem 68
211 W=PEEK(MP):IFW=0ORW=AORW=9THENGOSUB501:NEXT           :rem 106
213 L=L+1:GOTO71                                          :rem 162
501 POKE36877,200:FORT=15TO1STEP-.3:POKE36878,T:NEXT:FORT=1TO
    255:POKE36879,T:NEXT                                   :rem 79
502 POKE36877,0:POKE36878,15:POKE36879,8:M=M+1:TE=0:IFM=3THEN
    5000                                                   :rem 65
503 MP=8141:F=100:TE=0:RETURN                             :rem 232
3000 TE=TE+1:IFTE>3THENRETURN                             :rem 112
3001 FORPZ=MPTO7800STEP-22:IFPEEK(PZ)=2THENS=S+20*L       :rem 219
3002 POKEPZ,5:POKEPZ+C,7:POKEPZ,J:NEXT:FORT=180TO254STEP7:POK
     E36875,T:NEXT:POKE36875,0:RETURN                      :rem 60
4000 DATA0,0,0,255,0,0,0,0,0,0,84,56,124,56,84,0,60,126,231,2
     39,231,239,110,60,56,84                               :rem 33
```

```
4001 DATA108,186,146,170,170,16,42,62,20,20,8,8,0,0,16,0,16,0
     ,16,0,16,0,153,189,126                                  :rem 251
4002 DATA189,153,165,165,153,0,0,28,62,28,8,8,8,8,8,73,93,93,
     93,107,73,170,85,170,85,170                             :rem 62
4003 DATA85,170,0,0,0,0,0,0,0,0,0,0                          :rem 78
4004 DATA169,128,141,19,145,169,0,133,1,133,2,169,127,141,34,
     145,162,119,236,32,145                                  :rem 39
4005 DATA208,4,169,1,133,1,169,255,141,34,145,162,118,236,17,
     145,208,4,169,22,133,1                                  :rem 35
4006 DATA162,110,236,17,145,208,4,169,1,133,2,162,122,236,17,
     145,208,4,169,22,133,2,96                               :rem 177
5000 PRINT"{CLR}":POKE36869,240:PRINT"{HOME}{3 DOWN}";:X=(S<1
     00ANDS>=0)                                              :rem 1
5001 X1=(S>=100ANDS<300):X2=(S>=300ANDS<600):X3=(S>=600ANDS<1
     000):X4=(S>=1000)                                       :rem 153
5002 IFX1THEND$="{RED}SPACE CADET"                           :rem 224
5003 IFX2THEND$="{CYN}CAPTAIN"                               :rem 152
5004 IFX3THEND$="{BLU}WARRIOR"                               :rem 64
5005 IFX4THEND$="{GRN}TRUE EX-BLASTER!{YEL}THE BEST"         :rem 0
5006 IFXTHEND$="LOUSY PILOT!!"                               :rem 48
5007 PRINT"YOU ARE A ";D$:PRINT"{PUR}YOUR {RED}SCORE{YEL} WAS
     ";S                                                     :rem 251
```

Jim Koermer
and Bo Short

Meteors

"Meteors" is for the unexpanded VIC and requires a joystick.

Maneuver your spaceship at the bottom of the screen (represented by the spade) to avoid a collision with a deadly meteor. Any contact with a meteor means instant destruction for your spaceship.

Destroy the meteors with your laser before they reach your ship. But be careful—only the solid meteors can be destroyed by your laser. There is no way to destroy the others.

You'll be rewarded with 15 points for each meteor destroyed, and you have just two spaceships with which you can do battle. After the game is over, your score and the high score will be displayed.

Meteors

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 HI=0                                                    :rem 95
20 A=8086:C=7778:D=7770:E=7916:K=7832:H=7948:J=8002:L=81:M=32
   :N=44:Q=22:SC=0:LI=2                                    :rem 85
30 POKE36879,106:PRINT"{CLR}{WHT}":GOSUB410                :rem 88
50 POKEA,65:POKEC,L:POKED,L:POKEE,L:POKEF,L:POKEH,L:POKEJ,87
                                                          :rem 121
60 DD=37154:P1=37151:P2=37152                              :rem 35
70 POKEDD,127:P=PEEK(P2)AND128:J0=-(P=0):POKEDD,255:P=PEEK(P1
   )                                                      :rem 180
80 J1=-((PAND8)=0):J2=-((PAND16)=0):J3=-((PAND4)=0):FB=-((PAN
   D32)=0)                                                :rem 186
90 X=0:IFJ0THENX=2                                        :rem 103
100 IFJ2THENX=-2                                          :rem 191
110 IFFBTHENGOSUB480                                      :rem 244
120 A1=A+X                                                  :rem 6
130 IFA1<8076THENA1=A                                     :rem 197
140 IFA1>8096THENA1=A                                     :rem 202
150 POKEA,M:POKEA1,65:A=A1                                 :rem 27
160 Z=INT(RND(1)*10)*2                                    :rem 183
170 IFC=ATHENGOSUB580                                      :rem 52
180 IFC>8075THENPOKEC,M:C=7768+Z                          :rem 178
190 POKEC,M:C=C+N:POKEC,L                                  :rem 31
200 Y=INT(RND(1)*10)*2                                    :rem 177
210 IFD=ATHENGOSUB580                                      :rem 48
220 IFD>8075THENPOKED,M:D=7768+Y                          :rem 175
230 POKED,M:D=D+N:POKED,L                                  :rem 30
240 W=INT(RND(1)*10)*2                                    :rem 179
250 IFE=ATHENGOSUB580                                      :rem 53
260 IFE>8075THENPOKEE,M:E=7768+W                          :rem 180
270 POKEE,M:E=E+N:POKEE,L                                  :rem 38
280 V=INT(RND(1)*10)*2                                    :rem 182
290 IFK=ATHENGOSUB580                                      :rem 63
```

```
300 IFK>8075THENPOKEK,M:K=7768+V                          :rem 192
310 POKEK,M:K=K+N:POKEK,L                                  :rem 57
320 T=INT(RND(1)*10)*2                                    :rem 175
330 IFH=ATHENGOSUB580                                      :rem 55
340 IFH>8075THENPOKEH,M:H=7768+T                          :rem 185
350 POKEH,M:H=H+N:POKEH,87                                 :rem 84
360 S=INT(RND(1)*10)*2                                    :rem 178
370 IFJ=ATHENGOSUB580                                      :rem 61
380 IFJ>8075THENPOKEJ,M:J=7768+S                          :rem 194
390 POKEJ,32:J=J+44:POKEJ,87:GOTO70                       :rem 108
410 FORI=7680TO7701:POKEI,120:NEXT:PRINT"{7 RIGHT}METEORS!"
                                                           :rem 73
430 FORI=7724TO7745:POKEI,121:NEXT:FORI=8120TO8141:POKEI,120:
    NEXT                                                   :rem 10
450 FORI=8164TO8185:POKEI,121:NEXT                         :rem 61
460 PRINT"{19 DOWN}SCORE 0{7 RIGHT}LIFE 2":RETURN         :rem 145
480 B=A-N:POKEB,93:POKEB-Q,93:POKEB-N,93:POKEB-66,93:POKEB-88
    ,93                                                    :rem 85
490 IFB=CORB-44=CORB-88=CTHENGOSUB550:C=7768+Z             :rem 74
500 IFB=DORB-44=DORB-88=DTHENGOSUB550:D=7768+Y             :rem 69
510 IFB=EORB-44=EORB-88=ETHENGOSUB550:E=7768+W             :rem 72
520 IFB=KORB-44=KORB-88=KTHENGOSUB550:K=7768+V             :rem 96
530 POKEB,M:POKEB-22,M:POKEB-N,M:POKEB-66,M:POKEB-88,M:POKEB+
    22,M:RETURN                                            :rem 33
550 POKE36878,15:POKE36877,159:FORI=1TO100:POKE36878,0
                                                          :rem 143
560 SC=SC+15:PRINT"{HOME}{21 DOWN}SCORE ";SC:RETURN       :rem 127
580 POKE36878,15:POKE36877,145:FORI=1TO500:NEXT:POKE36878,0:L
    I=LI-1                                                  :rem 9
590 PRINT"{HOME}{21 DOWN}{13 RIGHT}LIFE ";LI:IFLI>0THENRETURN
                                                           :rem 43
600 PRINT"{CLR}{11 DOWN}{6 RIGHT}GAME{RIGHT}OVER!":PRINT"
    {5 RIGHT}PLAY AGAIN ?"                                :rem 104
610 IFSC>HITHENHI=SC                                       :rem 30
620 PRINT"{4 UP}{4 RIGHT}HIGH SCORE";HI:PRINT"{6 RIGHT}SCORE
    {SPACE}";SC                                           :rem 143
640 GETA$:IFA$=""THEN640                                   :rem 87
650 IFA$="Y"THENGOTO20                                     :rem 51
660 PRINT"{CLR}{BLU}":POKE36879,27:END                     :rem 63
```

Robert
Lewis

Hyper Ballshot

"Hyper Ballshot" is a good old-fashioned target game for the unexpanded VIC. It is a two-player game using paddles.

Across the top of the screen you'll see each player's score, the time clock, and the level the player who is playing reached. Near the top of the game board is the area where the targets will appear. The blank areas are where the targets will appear one at a time. The target's level, up to ten, has its own color. These colors are red, cyan, purple, green, and blue. After blue, the sequence repeats again. The value for these targets is ten times the level number. However, the yellow targets are worth minus five times the level value. So accuracy counts a lot.

Each player takes a turns which lasts 90 seconds. Begin each round by pressing and releasing the fire button. When you do this, the clock resets to zero. Using the knob line up the "shooter" with the target. When you press the fire button, the top and side bars turn orange and a ball flies into the target.

When you type in the program, be extra careful of lines 1000 and greater. All DATA statements, except the last one, have nine numbers. The last one has ten.

When you run the program, if you have an error in lines less than 1000, press RUN/STOP–RESTORE, correct the mistake, and restart the program by typing RUN2. If you just run the program, you will get an OUT OF MEMORY message. Be sure to use the integer arrays as shown. Not typing the percent (%) sign increases the memory needed for the arrays, and the program will not run.

Hyper Ballshot

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 GOSUB200:GOTO1000                                        :rem 119
2 GOSUB200                                                  :rem 68
4 DIMW%(8,5,1),H%(16),S(1):V=36878:GOSUB700:POKEV-9,255:POKEV
  ,47:POKEV+1,9:GOSUB300                                    :rem 97
5 H=0:B=2:T=38864:MM=T+7                                   :rem 116
30 FORI=1TO15STEP2:H%(I)=1:H%(I+1)=-1:NEXT:POKEV,47         :rem 46
35 TI$="000130":PR=0:KK=7                                   :rem 91
40 M=M-SGN(M-(15-(INT(PEEK(36872+PR)/18.2))))             :rem 209
55 K=T+M:POKEMM+22,0:POKEK+22,9:MM=K                       :rem 176
60 GOSUB600:IFF(PR)THENPOKEV,143:GOSUB500:POKEV,63         :rem 153
70 PRINT"{HOME}{WHT}"TAB(13);"TIME";MID$(TI$,4,1);":";RIGHT$(
   TI$,2):IFF(PR)=0ANDHTHEN80                              :rem 185
75 PRINT"{HOME}SCORE#1{5 SPACES}{5 LEFT}";S(0):PRINT"{HOME}
   {2 DOWN}SCORE#2{5 SPACES}{5 LEFT}";S(1)                 :rem 114
76 PRINT"{HOME}{2 DOWN}";TAB(13);"LEVEL#";B-1+BB:H=-1:rem 251
```

```
80 IFTI$="000130"THENGOSUB800:GOSUB790                    :rem 254
90 GOTO40                                                   :rem 6
200 PRINT"{CLR}{BLU}"TAB(4)"HYPER BALLSHOT"               :rem 150
203 PRINTTAB(9);"{DOWN}BY:":PRINTTAB(5);"{DOWN}ROBERT LEWIS"
                                                          :rem 238
205 PRINT"{4 DOWN}YOU WILL NEED PADDLES"                  :rem 250
210 PRINT"{2 DOWN}{RVS}STAND BY FOR THE GAME{8 SPACES}TO BEGI
    N{7 SPACES}"                                          :rem 151
220 RETURN                                                :rem 116
300 GOSUB390:PRINT"{CLR}{5 DOWN}{RVS}[*]{OFF}{2 SPACES}";:T=3
    2:GOSUB380:PRINT"{2 SPACES}{RVS}£{OFF}":PRINT"{RVS} [*]
    {OFF} ";:GOSUB380                                      :rem 91
310 PRINT" {RVS}£ {OFF}":PRINT"{BLU}{RVS}{2 SPACES}{OFF}*";:
    T=86:GOSUB380:PRINT"*{RVS}{2 SPACES}{OFF}":PRINT"{RVS}
    {2 SPACES}{OFF}>{WHT}";:T=94:GOSUB380                  :rem 6
320 PRINT"{BLU}<{RVS}{2 SPACES}{OFF}":PRINT"{RVS}{2 SPACES}
    {OFF}>";:FORB=1TO7:PRINT"{YEL}£{BLK}]";:NEXT:PRINT"{YEL}
    Y{BLU}<{RVS}{2 SPACES}{OFF}"                          :rem 226
330 GOSUB390:PRINT"{RVS}{2 SPACES}{OFF}@{BLK}";:T=90:FORH=1TO
    4:FORPR=1TO3:GOSUB380                                 :rem 170
340 GOSUB390:PRINT"@{RVS}{2 SPACES}{OFF}":PRINT"{RVS}
    {2 SPACES}{OFF}@{BLK}";:NEXTPR:T=T-1:NEXTH            :rem 181
350 T=61:GOSUB380:GOSUB390:PRINT"@{RVS}{2 SPACES}{OFF}{WHT}
    {HOME}"                                                :rem 66
360 RETURN                                                :rem 121
380 FORB=1TO15:PRINTCHR$(T);:NEXT:RETURN                  :rem 118
390 POKE646,9:RETURN                                      :rem 234
500 POKEK,7:FORY=1TO11:POKEK-(Y*22)+22,0:POKEK-(Y*22),7:POKEV
    -2,240-Y*2:NEXTY:J=K-Y*22                             :rem 147
505 IF-(H%(M))THENS(PR)=S(PR)-5*(B-1+BB):POKEV-3,128 :rem 188
510 POKEV-2,0:POKEJ+22,0:POKEV-3,0:IFH%(M)THENRETURN :rem 210
515 IFM/2=INT(M/2)THENPOKEJ,0:S(PR)=S(PR)+10*(B-1+BB):H%(M)=-
    1:X=X+1:POKEV-3,255                                   :rem 166
520 IFX=7THENX=0:B=B+1:IFB=7THENB=2:BB=5                  :rem 204
530 GOSUB790                                              :rem 184
540 POKEV-3,0:RETURN                                      :rem 244
600 POKE37154,127:F(1)=-((PEEK(37152)AND128)=0):POKE37154,255
                                                           :rem 37
610 F(0)=-((PEEK(37137)AND16)=0):RETURN                   :rem 166
700 FORMM=0TO1:FORB=0TO5:W%(0,B,MM)=INT(RND(1)*7)+1:NEXTB,MM
                                                          :rem 137
710 FORMM=0TO1:FORB=0TO5:FORT=1TO6:POKEV+1,25+T           :rem 235
720 H=INT(RND(1)*7)+1                                     :rem 125
730 FORPR=0TO7                                            :rem 106
740 IFW%(PR,B,MM)=HTHEN720                                 :rem 26
750 NEXTPR:W%(T,B,MM)=H:NEXTT,B,MM                        :rem 146
760 RETURN                                                :rem 125
790 POKET-12*22+W%(X,B-2,PR)*2,B:H%(W%(X,B-2,PR)*2)=0:RETURN
                                                          :rem 145
800 POKET-12*22+W%(X,B-2,PR)*2,0:H%(W%(X,B-2,PR)*2)=-1:POKEV,
    47                                                    :rem 225
```

```
804 FORQ=1TO100:POKEV-3,255-Q:NEXT:POKEV-3,0          :rem 207
805 IF-PRANDTI$=>"000130"THENPRINT"{HOME}{4 DOWN}{WHT}ANOTHER
     GAME?{RVS}(Y OR{SHIFT-SPACE}N){OFF}":GOTO950      :rem 38
810 X=0:B=2:IFS(0)<>0THENPR=1:BB=0:H=0                :rem 233
820 GOSUB600:IFF(PR)=0THEN820                         :rem 232
830 GOSUB600:IFF(PR)THEN830                           :rem 125
840 TI$="000000":POKEV,95:RETURN                      :rem 113
910 NEXT:END                                          :rem 234
950 GETA$:IFA$=""OR(A$<>"Y"ANDA$<>"N")THEN950         :rem 17
960 IFA$="Y"THENPOKEV-9,240:RUN2                      :rem 170
970 POKEV-9,240:POKEV+1,27:PRINT"{CLR}{BLU}GOODBYE!!!!!":POKE
    56,PEEK(56)+2:END                                 :rem 183
1000 X=PEEK(56)-2:POKE52,X:POKE56,X:POKE51,PEEK(55):CLR
                                                      :rem 56
1010 CS=256*PEEK(52)+PEEK(51)                         :rem 64
1020 FORI=CSTOCS+511:POKEI,PEEK(I+33792-CS):NEXT      :rem 205
1030 READX:IFX=0THEN2                                 :rem 39
1040 FORI=XTOX+7:READY:POKEI,Y:NEXT                   :rem 70
1050 GOTO1030                                         :rem 195
1100 DATA7352,28,50,125,127,127,127,62,28             :rem 20
1110 DATA7360,0,28,54,58,62,62,28,0                   :rem 231
1120 DATA7368,0,0,24,52,60,60,24,0                    :rem 165
1130 DATA7376,0,0,16,40,56,16,0,0                     :rem 115
1140 DATA7392,255,165,231,153,153,239,165,255         :rem 235
1150 DATA7400,126,102,90,102,102,90,102,126           :rem 92
1160 DATA7408,0,0,60,126,165,231,189,129              :rem 228
1170 DATA7168,190,190,190,190,190,190,190,190,0       :rem 69
```

Robert L.
Lykins

Laser Command

Scout ships from the planet Zardon are orbiting our planet to determine the chances of a successful invasion. Your mission, as First Master Gunner of Laser Defense Post One, is to destroy as many of the scout ships as possible. Your commander thus hopes to prove to the Zardon Empire that the risks of invasion are too great. Will you be able to down enough scouts alone? The commander thinks not and assigns Second Master Gunner to Laser Defense Post Two. Now you must prove you deserve the title of First Master Gunner by scoring the most hits.

This exciting game requires the *Super Expander* cartridge and paddle controllers. If you had thought the *Super Expander* was merely for drawing graphs and pictures, then look again. These extra graphics capabilities can greatly enhance game play. In this game, the paddle controllers are used to direct laser beams at precise angles on the screen.

Special Features

When you run "Laser Command," it first checks to see if you have inserted the *Super Expander* by PEEKing memory location 56 (top of RAM). The value found there should be 29 if the graphics screen has not yet been set up with a GRAPHIC command.

To set up the graphics screen, the *Super Expander* sets aside over 3K of memory for bitmapping a 20 × 20 space area. The top-of-RAM pointer is set to 16. Using the GRAPHIC4 command will reset the pointer to 29. This is why you seem to lose over half the memory indicated on cold start after using GRAPHIC1, GRAPHIC2, or GRAPHIC3. You actually do lose it to the graphics screen and should bear this in mind when writing programs. This program uses all but about 30 bytes in execution, so it would be wise not to add unnecessary spaces when typing it in.

Beginning Play

After pressing a controller fire button, you may center the screen with the cursor keys. You then have the option of playing against the computer or another player. If you choose to play against your VIC, be aware that it will prove to be a formidable opponent. The better you play, the harder the computer tries to beat you. This automatic level control is programmed in line 600.

Keep practicing: Experience will help you beat the VIC!

Laser Command

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
105 IFPEEK(56)=29ORPEEK(56)=16THEN115:REM CHECKS FOR SUPER EX
    PANDER                                                  :rem 54
110 PRINT"{CLR}{RVS}{RED}REQUIRES SUPEREXPANDERINSERT AND REL
    OAD GAME":END                                            :rem 4
115 PRINT"{CLR}{3 DOWN}{RVS}{BLU}REQUIRES PADDLE CON-
    {2 SPACES}TROLLERS. PUSH A FIRE{3 SPACES}BUTTON TO START.
    {4 SPACES}"                                             :rem 13
120 IFRJOY(0)=0THEN120                                      :rem 22
125 GOSUB540:Z=RND(-TI):REM RANDOMIZE                       :rem 72
130 GRAPHIC2:COLOR14,4,0,0                                 :rem 250
135 CHAR0,9,"PO":CHAR19,9,"@L":CHAR9,0,"L":CHAR9,19,"@":CHAR1
    0,0,"O":CHAR10,19,"P                                   :rem 179
140 REGION2:CHAR3,1,"Z LASER{2 SPACES}COMMAND Z            :rem 235
145 REGION1:CHAR6,2,"*USE CURSOR KEYS":CHAR8,2,"TO CENTER SCR
    EEN                                                    :rem 124
150 REGION4:CHAR11,4,"CHOOSE ONE:                          :rem 211
155 REGION1:CHAR14,0,"1. YOU VS. COMPUTER":CHAR16,0,"2. 2 PLA
    YER/PRACTICE                                           :rem 253
160 GETA$:REM SCREEN CENTERING AND MENU SELECTION            :rem 7
165 IFA$="{RIGHT}"THENPOKE36864,PEEK(36864)+1:GOSUB540:IFPEEK
    (36864)=12THENPOKE36864,3                              :rem 165
170 IFA$="{DOWN}"THENPOKE36865,PEEK(36865)+1:GOSUB540:IFPEEK(
    36865)=44THENPOKE36865,20                              :rem 205
175 IFA$<>"1"ANDA$<>"2"THEN160                             :rem 110
180 POKE785,VAL(A$):REM STORES SELECTION FOR FUTURE REFERENCE
                                                            :rem 38
185 SCNCLR:COLOR6,2,1,0                                    :rem 105
190 CHAR1,0,"{2 SPACES}BLAST THE UFOS                       :rem 39
195 CHAR3,0,"THAT FLY ABOVE THE                              :rem 4
200 CHAR5,0,"LASER DEFENSE POSTS.                          :rem 199
205 CHAR7,0,"YOU GAIN 100 POINTS                            :rem 32
210 CHAR9,0,"FOR EACH HIT BUT                               :rem 92
215 CHAR11,0,"LOSE 10 POINTS FOR                            :rem 26
220 CHAR13,0,"EACH MISS.                                    :rem 59
225 CHAR15,0,"TIME LIMIT: 3 MIN.                           :rem 244
230 REGIONRND(1)*8:CHAR18,0,"*PRESS FIRE BUTTON.*          :rem 157
235 GOSUB540                                               :rem 179
240 IFRJOY(0)=0THEN230                                      :rem 27
245 GOSUB540                                               :rem 180
250 SCNCLR:CLR:COLOR0,6,0,0:GOSUB550:REM SCREEN AND VARIABLE
    {SPACE}SET-UP                                          :rem 131
255 SX=1000:SY=SX:B=6:S$=" ":U$="Z":S=1:H=PEEK(36864):V=PEEK(
    36865):P=PEEK(785)                                     :rem 173
260 GOSUB580:GOSUB585                                       :rem 17
265 REM DRAW DEFENSE POSTS                                  :rem 66
270 REGION5                                                :rem 146
275 DRAW2,311,1023TO361,835TO411,1023:PAINT2,361,900        :rem 88
280 REGION4                                                :rem 146
```

```
285 DRAW2,612,1023TO662,835TO712,1023:PAINT2,662,900 :rem 105
290 TI$="000000                                      :rem 219
295 IFSX=0ORSY=0THEN485                              :rem 209
300 RJ=RJOY(0)/4:IFTI$>"000300"THEN485               :rem 147
305 REM FIRING ROUTINES                              :rem 180
310 X=RPOT(0)*4:IFRDOT(X,PZ)<>0AND(RJ=1ORRJ=3)THENXZ=1
                                                     :rem 162
315 Y=RPOT(1)*4:IFRDOT(Y,PZ)<>0AND(RJ=2ORRJ=3)THENYZ=1
                                                     :rem 172
320 REGION1:SOUND0,(RJAND1)*230,0,0,3:IFRJ=0ORRJ=2THENGOSUB54
    0:GOTO350                                        :rem 65
325 DRAW2,361,815TOX,PZ                              :rem 91
330 DRAW0,361,815TOX,PZ                              :rem 85
335 IFRJAND1THENSX=SX-10                             :rem 26
340 IFXZ=1THENGOSUB415                               :rem 143
345 XZ=0:GOSUB580                                    :rem 18
350 REGION1:SOUND0,0,(RJAND2)*101,0,3:IFP=1THEN595   :rem 202
355 IFRJ=0ORRJ=1THENGOSUB540:GOTO385                 :rem 59
360 DRAW2,662,815TOY,PZ                              :rem 95
365 DRAW0,662,815TOY,PZ                              :rem 98
370 IFRJAND2THENSY=SY-10                             :rem 28
375 IFYZ=1THENGOSUB415                               :rem 152
380 YZ=0:GOSUB585                                    :rem 23
385 REGIONINT(RND(1)*6+2)                            :rem 195
390 REM UFO MOVEMENT                                 :rem 213
395 CHARZ,Q,S$:Q=Q+S:IFQ=20ORQ=-1THENGOSUB550        :rem 95
400 CHARZ,Q,U$                                       :rem 46
405 GOTO295                                          :rem 114
410 REM EXPLOSION SUBROUTINE                         :rem 74
415 IFTI$>"000230"THENB=2                            :rem 113
420 CHARZ,Q,"V                                       :rem 175
425 FORL=9TO0STEP-1                                  :rem 177
430 POKE36864,H+1:POKE36865,V+L:IFL=8THEN:COLOR1,7,0,0
                                                     :rem 205
435 POKE36864,H:POKE36865,V:IFL=7ANDXZ=1THEN:COLOR6,5,0,0
                                                     :rem 244
440 POKE36864,H-1:POKE36865,V-L:IFL=7ANDYZ=1THEN:COLOR6,4,0,0
                                                     :rem 199
445 POKE36864,H:POKE36865,V:IFL=4THEN:COLOR0,B,0,0   :rem 6
450 SOUND0,0,0,175+L*5,L                             :rem 33
455 NEXT:CHARZ,Q,S$:GOSUB550                         :rem 3
460 REM SCORING                                      :rem 147
465 IFXZ=1THENSX=SX+110                              :rem 205
470 IFYZ=1THENSY=SY+110                              :rem 204
475 RETURN                                           :rem 128
480 REM END OF GAME                                  :rem 6
485 FORL=1TO500:NEXT:SOUND0,0,0,0,0                  :rem 86
490 REGION7                                          :rem 152
495 CHAR5,0,"REPLAY - PRESS FIRE                     :rem 76
500 CHAR7,0,"RESELECT - HIT SHIFT                    :rem 123
505 CHAR9,0,"QUIT - PRESS SPACE                      :rem 4
```

```
510 FORL=1TO999:NEXT                                        :rem 254
515 COLOR(VAL(RIGHT$(TI$,2))+2)/4,VAL(MID$(TI$,5,1))+2,0,0
                                                            :rem 180
520 IFRJOY(0)<>0THENFORL=1TO150:NEXT:GOTO245                 :rem 37
525 IFPEEK(653)=1THENCLR:SCNCLR:GOTO130                     :rem 195
530 IFPEEK(197)=32THEN:GRAPHIC4:POKE36879,27:PRINT"{CLR}{BLU}
    ":GETA$:END                                             :rem 215
535 GOTO515                                                 :rem 113
540 FORL=1TO75:NEXT:RETURN:REM MULTI-PURPOSEDELAY            :rem 79
545 REM UFO DIRECTION AND ALTITUDE                           :rem 60
550 Z=INT(RND(1)*8+5):PZ=Z*(55-Z*.1):REM LASER ALTITUDE FACTO
    R                                                       :rem 181
555 S=-S                                                    :rem 175
560 IFS<0THENQ=19                                            :rem 16
565 IFS>0THENQ=0                                            :rem 221
570 RETURN                                                  :rem 124
575 REM DISPLAY SCORE                                        :rem 23
580 REGION5:CHAR1,0,STR$(SX)+S$:RETURN                      :rem 124
585 REGION4:CHAR1,19-LEN(STR$(SY)),STR$(SY):RETURN          :rem 144
590 REM COMPUTER FIRE CONTROL SUBROUTINE                     :rem 72
595 Y=INT(RND(1)*1024)                                      :rem 204
600 IFRND(1)<SX*.0003THENY=Q*(55-Q*.1)                      :rem 143
605 IFRDOT(Y,PZ)<>0THENYZ=1                                 :rem 221
610 SY=SY-10:SOUND0,0,202,0,3                               :rem 132
615 GOTO360                                                 :rem 110
```

Joe W.
Rocke

Space Blockade

Warp-speed action (well, almost) and an ever-changing kaleidoscope of color graphics are only part of the unusual game features of "Space Blockade" for the unexpanded VIC.

Running the Blockade

The game field consists of a bordered playing screen that begins to fill with randomly located alien symbols. Meanwhile, blockading bricks also begin appearing. A white spaceship character, which serves as a shooter, is located at the top of the screen. The shooter can be pointed up, down, right, or left and is moved by using the joystick. The aliens serve as targets that are exploded by aiming the shooter and pressing the fire button. The bricks act as barriers that stop shooter movement. The object of the game is to blast as many of the alien symbols as you can before being blockaded by the bricks.

If you chose a play level of 6 from the scale of 1 to 9, the shooter will be stationary. Moving the joystick in any one of the four primary directions points the shooter accordingly. If the joystick is held in that position, the shooter will move slowly. Press the fire button and the shooter takes off at super speed until it is blocked by a brick or the wall. If the shooter was aimed at a target, the fire button must be pressed to explode the target and score.

The action is faster if the shooter is aimed at a target and the fire button pressed. In this case the shooter zips to the target and obliterates it. The current game score, denoted by an S at the top of the screen, updates to score the hit. Scoring values vary based on the color of the target. The high score is denoted by an H at the top of the screen.

Blue checkered bricks appear on the game field at random locations as new targets continue to show up. The bricks block the shooter's movement, forcing you to maneuver the shooter. Whenever you decide it is time to give up, the play can be ended by pressing the Y key.

Easy or Hard

You can choose a level of difficulty ranging from 1 (fast) to 9 (slow). Once you've chosen the level, the screen will blank for a few seconds while data for the custom graphics is POKEd to a protected memory location. The bordered game field then appears with the shooter located at top center of the border.

At levels 1 to 5, the shooter will alternately rotate clockwise while remaining at its initial position. The speed of rotation is dependent upon the level of difficulty chosen. You must coordinate joystick action with the shooter direction in order to move the shooter. Of course, this makes the play more challenging and the scoring more difficult.

Play levels of 6 to 9 make the shooter easier to control. Within this range the shooter is fixed in its initial position until rotated and moved by the joystick. Shooter movement is relatively slow if the joystick is held in the direction of movement position. However, if the fire button is pressed when the shooter is pointed at a target, the shooter will zip to the target fast as a speeding bullet.

Space Blockade

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 POKE51,0:POKE52,28:POKE55,0:POKE56,28:CLR:DIMJ(2,2):POKE37
   139,0:POKE36879,93                                  :rem 137
15 PRINT"{CLR}"                                        :rem 202
20 JA=37137:JB=37152:JC=37154:H=0:N=0:FORM=256TO274:READD:POK
   EM,D:NEXT                                           :rem 225
30 SYS256:POKE36878,15:FORM=7432TO7551:READD:POKEM,D:NEXT
                                                        :rem 51
40 FORL=0TO2:FORK=0TO2:READJ(K,L):NEXTK,L              :rem 119
50 POKE36869,255:S$="*)*,**,**,**,**,**,*)*":M=7680:C=30720
                                                       :rem 147
60 PRINT"{CLR}{CYN}";:POKE198,0:POKE36879,8:FORX=0TO21:PRINTS
   $;:NEXT                                             :rem 199
70 PRINT"{HOME}{2 DOWN}"SPC(6)"{10 SPACES}";SPC(12)"{PUR} BLO
   CKADE ";                                             :rem 67
80 PRINTSPC(12)"{10 SPACES}{2 DOWN}"SPC(9)"{YEL}BY: JOE W. RO
   CKE"                                                :rem 243
90 PRINT"{4 DOWN}{2 RIGHT}SELECT{2 SPACES}DIFFICULTY{DOWN}";S
   PC(10)"1 {RVS}--{OFF} 9"                            :rem 251
100 GETA$:IFA$>"9"ORA$<"1"THEN100                       :rem 54
110 E=VAL(A$):IFE>5THENN=5:E=E-5                       :rem 128
120 S=0:R=0:PRINT"{CLR}":POKE36879,8:FORL=M+23TOM+42:POKEL,39
    :POKEL+462,37:POKEL+C+462,3                        :rem 120
130 POKEL+C,3:NEXT:FORL=M+44TOM+462STEP22:POKEL,38:POKEL+21,4
    0:POKEL+C,3:POKEL+C+21,3                           :rem 231
135 NEXT                                               :rem 216
140 T=RND(-TI):T=M+INT(RND(1)*20)+45:W=35:POKET,W:POKE646,3:P
    RINT"{HOME}{GRN}"SPC(12)"H";H                      :rem 237
150 FORL=1TOE:GOSUB330:U=J(X+1,V+1):Y=T+U:IFPEEK(Y)=32THENPOK
    ET,32:T=Y                                          :rem 173
160 IFFBTHENGOSUB360                                   :rem 246
170 NEXTL:IFN<>5THEN220                                :rem 172
180 W=W-1:IFU=22THENW=34                               :rem 203
190 IFU=-22THENW=32                                    :rem 116
200 IFU=-1THENW=35                                      :rem 60
210 IFU=1THENW=33                                       :rem 14
220 IFN=9THENGOSUB360:IFR>200THEN450                   :rem 226
230 A=INT(RND(1)*6):W=W+1:IFW>36THENW=33               :rem 134
240 IFA=0THENA=7                                       :rem 183
```

```
250 PRINT"{HOME}{RIGHT}{YEL}S"S:POKET+C,A:POKET,W:B=M+INT(RND
    (1)*440)+44:U=INT(RND(1)*3)+41                          :rem 55
260 POKET+C,1:B=B+1:IFB>8162THENB=7725                     :rem 218
270 IFPEEK(B)=32THENPOKEB,U:POKE36874,100+U:POKEB+C,A:GOTO310
                                                           :rem 226
280 IFPEEK(B)=45THEN260                                     :rem 78
290 IFPEEK(B)<41THEN260                                     :rem 74
300 POKEB,45:POKEB+C,3:R=R+1:IFR=398THEN450                 :rem 76
310 POKE36874,0:IFPEEK(197)=11THEN450                      :rem 114
320 GOTO150                                                :rem 100
330 POKEJC,127:S3=-((PEEK(JB)AND128)=0):POKEJC,255         :rem 127
340 P=PEEK(JA):S1=-((PAND8)=0):S2=((PAND16)=0):S0=((PAND4)=0)
                                                           :rem 172
350 FB=-((PAND32)=0):X=S2+S3:V=S0+S1:RETURN                 :rem 63
360 IFW=33THEND=-22                                         :rem 99
370 IFW=34THEND=1                                            :rem 5
380 IFW=35THEND=22                                          :rem 58
390 IFW=36THEND=-1                                          :rem 54
400 Y=T+D:IFPEEK(Y)=32THENPOKET,32:POKEY,W:T=Y:GOTO400
                                                           :rem 218
410 IFPEEK(Y)<41ORPEEK(Y)=45THENRETURN                     :rem 185
420 S=S+PEEK(Y):POKEY,44:POKET,32:T=Y                       :rem 59
430 FORL=200TO220:POKE36877,L:NEXT:POKE36877,255:FORL=1TO200:
    NEXT:POKE36877,0:POKET,W                                :rem 22
440 RETURN                                                 :rem 120
450 POKE36869,240:POKE36879,253                            :rem 218
460 N=0:PRINT"{CLR}{BLK}{RVS}{2 DOWN}"SPC(7)"GAME OVER":PRINT
    "{4 DOWN}{RED} SCORE"S:IFS>HTHENH=S                    :rem 162
470 PRINT"{2 DOWN}{BLU} HIGH SCORE"H:PRINT"{4 DOWN}{GRN}
    {3 SPACES}PLAY AGAIN (Y/N)"                            :rem 187
480 FORL=1TO1200:IFPEEK(197)=11THEN50                      :rem 182
490 IFPEEK(197)=28THEN510                                  :rem 175
500 NEXTL:N=9:POKE36869,255:GOTO120                        :rem 101
510 POKE36869,240:POKE36879,27:PRINT"{CLR}{BLU}":END       :rem 116
520 DATA 162,0,189,0,128,157,0,28,189,0,129,157,0,29,232,208,
    241,96,0                                                :rem 67
530 DATA 56,124,56,16,84,124,84,68,0,242,39,127,39,242,0,0
                                                            :rem 41
540 DATA 68,84,124,84,16,56,124,56,0,79,228,254,228,79,0,0
                                                            :rem 59
550 DATA 255,0,102,0,0,0,0,0,1,5,5,1,1,5,5,1                :rem 47
560 DATA 0,0,0,0,0,102,0,255,128,160,160,128,128,160,160,128
                                                            :rem 96
570 DATA 0,24,60,102,255,60,24,0,0,56,108,198,214,198,108,56
                                                           :rem 132
580 DATA 24,60,36,255,36,36,60,24,40,129,36,18,64,10,64,17
                                                            :rem 41
590 DATA 170,85,170,85,170,85,170,85,0,0,0,0,0,0,0,0,0,0,0 :rem 224
600 DATA 0,0,0,0,0,0,0,0                                   :rem 100
610 DATA-23,-22,-21,-1,0,1,21,22,23                        :rem 175
```

Richard
Ruef

Torpedo-8

"Torpedo-8" is a simple, yet enjoyable, air defense game for the unexpanded VIC. Your job, as captain of the fleet of seven bomber planes, is to destroy as many of the enemy ships as possible. Maneuver your planes while avoiding enemy fire, and shoot when you think you can land a hit.

Use the Z and slash (/) keys to move your plane to avoid the enemy ship fire. When you think you can make a hit, press the space bar and drop a bomb on a ship. When a plane is hit, it will go up in flames and quickly descend to the ocean.

Torpedo-8

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
4 PRINT"{CLR}{BLK}{6 DOWN}{6 RIGHT}TORPEDO-8"          :rem 190
6 PRINT"{2 DOWN} HIT (Z) TO GO LEFT":PRINT" HIT (/) TO GO RIG
  HT":PRINT" HIT (SPACE) TO BOMB"                      :rem 180
7 PRINT:PRINT" HIT ANY KEY TO START":FORO=1TO100:NEXT:RESTORE
                                                       :rem 159
8 POKE36879,25:P%=7:POKE650,128:U=7882                 :rem 158
9 T=0:PL%=7:DF%=166:GETD$:IFD$=""THEN9                 :rem 200
10 S=INT(RND(1)*12)+2:Z=100:GOSUB15:GOSUB14             :rem 30
11 REM***CHANGE PL% TO LENGTHEN GAME**CHANGE DF% FOR DIFF. FA
   CTOR*****                                           :rem 165
12 GOTO20                                              :rem 254
13 REM****DRAW SEA*****                                :rem 201
14 PRINT"{HOME}{22 DOWN}{RVS}{BLU}IJIJIJIJIJIJIJIJIJIJI{OFF}
   {BLK}{HOME}":GOSUB55:RETURN                         :rem 234
15 POKE36878,15:POKE36875,200:L=0:PRINT"{CLR}{BLK}{5 DOWN}"TA
   B(P%)"{4 SPACES}- "                                 :rem 159
16 POKE36876,148:PRINTTAB(P%)"{3 SPACES}*+*"           :rem 112
18 PRINTTAB(P%)"{4 P}*{4 P}"                           :rem 121
19 PRINTTAB(P%)" {C}K {RVS}X{OFF} J{V}":RETURN         :rem 253
20 A=7682+S:B=30720:GOSUB21:GOTO32                     :rem 223
21 GETD$:IFD$=""THENRETURN                             :rem 108
22 IFPEEK(197)=32THENGOSUB400:RETURN                     :rem 9
27 IFP%=0THEN29                                        :rem 116
28 IFD$="Z"THENP%=P%-1:GOSUB350:Z=Z-5:L=L+1:RETURN      :rem 84
29 IFP%>11THEN31                                       :rem 162
30 IFD$="/"THENP%=P%+1:GOSUB300:Z=Z-5:L=L+1:RETURN      :rem 27
31 GOSUB15:GOSUB14:RETURN                              :rem 131
32 FORQ=440TO0STEP-22:C%=PEEK(A+Q):READA%:POKEA+Q+B,0:POKEA+Q
   ,A%                                                 :rem 135
33 FORY=1TOZ:NEXTY:IFC%<>32THENGOSUB60:GOSUB190         :rem 92
34 IFPL%=0THENQ=0:Y=Z:GOTO600                           :rem 68
35 POKEA+Q+B,1:Z=Z-5                                   :rem 171
36 IFPEEK(197)<>64THENGOSUB21:Q=Q-22                   :rem 175
37 IFPEEK(197)=64ANDL>0THENGOSUB15:GOSUB14             :rem 120
38 IFQ<440THEN40                                       :rem 178
```

```
39 POKEA+462,121:POKEA+462+B,0                                  :rem 157
40 NEXTQ:RESTORE                                                 :rem 82
50 GOTO10                                                       :rem 255
54 REM****SHIP ON WATER******                                    :rem 69
55 POKEA+462,121:POKEA+462+B,0:RETURN                           :rem 181
59 REM***FIRE FROM PLANE*****                                   :rem 108
60 POKEA+Q+B,2:POKEA+Q,127:GOSUB703:RETURN                      :rem 119
100 DATA46,46,46,46,126,126,126,101,101,101,116,116,117,117,9
    7,97                                                        :rem 170
110 DATA246,246,234,234,160,81,81,81,81,81                       :rem 70
132 POKEA+Q+B,1                                                  :rem 76
185 REM***PLANE SHOT DOWN*****                                  :rem 184
190 Z=60                                                        :rem 151
195 GOSUB700                                                    :rem 182
200 FORW=QTO540STEP22:PRINT"{HOME}{DOWN}{LEFT}{INST} ":POKE21
    8,158:POKE218,144                                            :rem 49
201 GOSUB14:POKE36878,15:POKE36875,128+(W/10)                    :rem 35
205 FORY=1TOZ:NEXTY:Z=-5                                         :rem 71
206 POKEA+Q,104:POKEA+Q+B,7                                      :rem 59
207 IFW>250THENPOKEA+Q+B,2                                       :rem 58
208 IFW>350THENPOKEA+Q+B,0                                       :rem 58
210 NEXTW:Q=0:POKE36876,0                                       :rem 244
211 GOSUB14                                                     :rem 121
220 POKE36879,42:FORY=1TO250:NEXT:POKE36879,25:POKE36875,0:GO
    SUB700                                                       :rem 49
230 FORY=1TO150:NEXT:PL%=PL%-1:RETURN                           :rem 102
290 REM***PLANE-RIGHT*****                                      :rem 234
300 PRINT"{CLR}{4 DOWN}"TAB(P%)"{BLK} M"                        :rem 121
305 PRINTTAB(P%)"{2 SPACES}M"                                   :rem 211
310 PRINTTAB(P%)"{2 SPACES}{C}MMN"                               :rem 38
315 PRINTTAB(P%)"{4 SPACES}{RVS}W{OFF}M"                         :rem 79
320 PRINTTAB(P%)"{5 SPACES}M"                                   :rem 208
325 PRINTTAB(P%)"{5 SPACES}{C}M"                                :rem 145
330 PRINTTAB(P%)"{7 SPACES}M"                                   :rem 209
331 IFPEEK(197)=64THENGOSUB15:GOSUB14                            :rem 24
335 GOSUB14:RETURN                                              :rem 154
345 REM***PLANE-LEFT******                                      :rem 194
350 PRINT"{CLR}{4 DOWN}"TAB(P%)"{BLK}{7 SPACES}N"               :rem 127
352 PRINTTAB(P%)"{6 SPACES}N"                                   :rem 214
355 PRINTTAB(P%)"{3 SPACES}MNN{V}"                               :rem 50
360 PRINTTAB(P%)"{3 SPACES}N{RVS}W{OFF}"                         :rem 80
365 PRINTTAB(P%)"{3 SPACES}N"                                   :rem 218
370 PRINTTAB(P%)"{2 SPACES}N{V}"                                :rem 148
375 PRINTTAB(P%)" N"                                            :rem 219
376 IFPEEK(197)=64THENGOSUB15:GOSUB14                            :rem 33
380 GOSUB14:RETURN                                              :rem 154
390 REM****BOMB ROUTINE****                                       :rem 22
400 GOSUB703:J=0:M=20:K=97:FORY=0TO264STEP22                    :rem 189
401 IFPEEK(U+P%+Y)<>32ANDQ>DF%THENGOSUB500:Y=264:GOTO420
                                                                :rem 188
402 POKE36877,243-(Y/9):POKE36878,15                            :rem 227
```

```
403 IFPEEK(U+P%+Y)=121THENGOSUB450:Y=264:GOTO420      :rem 163
404 POKEU+P%+Y,K:POKEU+P%+Y+B,7                       :rem 105
405 FORV=1TOM:NEXTV                                     :rem 3
406 POKEU+P%+Y,?58:POKEU+P%+Y+B,0                     :rem 134
407 IFY>220THENK=46:GOTO418                           :rem 134
408 IFY>110THENK=116:GOTO418                          :rem 179
409 IFY>154THENK=39:GOTO418                           :rem 144
410 IFY>66THENK=117                                   :rem 119
411 IFPEEK((U+P%+Y)-22)=58THENPOKE(U+P%+Y)-22,32      :rem 234
418 M=M-1:NEXTY                                       :rem 164
419 POKEU+P%+Y,88:POKEU+P%+Y+B,2:GOSUB700             :rem 224
420 GOSUB703:GOSUB15:GOSUB14:RETURN                     :rem 9
450 GOSUB703:FORK=1TO5                                :rem 101
455 POKEU+P%+Y,233:POKEU+P%+Y+B,2                     :rem 183
460 FORC=1TO30:NEXT                                   :rem 177
465 POKEU+P%+Y,223:POKEU+P%+Y+B,7                     :rem 188
470 FORC=1TO30:NEXTC,K                                :rem 108
471 POKEU+P%+Y,32:GOSUB700:Q=0                         :rem 30
475 T=T+500:RETURN                                     :rem 95
500 GOSUB703:POKEU+P%+Y,42                             :rem 35
505 FORO=1TO5                                          :rem 22
510 POKEU+P%+Y+B,2                                      :rem 9
520 FORC=1TO20:NEXT                                   :rem 173
525 POKEU+P%+Y+B,7                                     :rem 20
530 FORC=1TO20:NEXTC,O:GOSUB700                       :rem 189
535 Q=0:RETURN                                        :rem 117
550 REM****END OF GAME****                             :rem 84
600 PRINT"{CLR}{6 DOWN}{6 RIGHT}GAME OVER":PRINT:PRINT
                                                      :rem 242
602 PRINT" YOU LOST 7 PLANES":PRINT:PRINT" YOU SCORED"T"PTS."
                                                      :rem 238
650 PRINT:PRINT:PRINT:PRINT"{5 SPACES}PLAY AGAIN?"    :rem 150
655 PRINT:PRINT"{4 SPACES}HIT (Y) OR (N)"               :rem 7
660 GETD$                                             :rem 228
665 IFD$="N"THENPOKE650,0:END                         :rem 205
670 IFD$="Y"THEN680                                    :rem 59
675 GOTO660                                           :rem 119
680 FORO=1TO10:GETY$:NEXT:GOTO1                       :rem 250
690 REM***SOUND EFFECT****                            :rem 223
700 POKE36877,128                                     :rem 156
701 FORO=15TO0STEP-1:POKE36878,O:GETD$                 :rem 84
702 FORR=1TO40:NEXTR,O                                :rem 141
703 FORC=36874TO36878:POKEC,0:NEXT:RETURN              :rem 91
```

Jim
Schmitz

Sevicog

So there you are, drawing another smiling face on your television screen with your *Super Expander* cartridge, impressing yet another relative. But, really, what can you use the *Super Expander* for? If you're like me, you may have realized that instead of using all the great built-in features of the *Super Expander,* you have ended up using it only for the extra memory. This, therefore, became the big question: How can I utilize the full potential of the VIC *Super Expander?*

The answer was "Sevicog," a game in which you test a new top-secret antimissile defense system code-named SEVICOG, which is embedded in a miniscule asteroid beyond the orbit of Mars. This test will measure the effectiveness of such a system before it is installed on Earth.

Several features were incorporated into Sevicog to make the game more interesting. First, you can input the number of targets you would like to descend down the screen at one time. Second, as the score increases, your radiation cloud disperses more slowly, and the missiles reappear lower and lower on the screen. As all of this goes on, you will find it getting harder to move quickly. However, to overcome this penalty, you can hold the button of the joystick down as you push the joystick in the desired direction to activate a quick-move feature. Pressing the fire button alone will explode your radiation bomb.

Now, before you type in the game itself, here are a few words about the program. For one, you obviously must use the *Super Expander.* Second, integer variables and the period are used to represent constants and zero, respectively. This greatly increases the speed of the program. Finally, there is some memory left if you want to make your own changes. You might want to change the equation for the falling missiles (line 10) from a line to a parabola or even a trigonometric equation. Also, you can try to make an even more dramatic ending or add an option to allow multiple firings.

To play the game, move your bomb under the falling missiles and press the fire button to explode the radiation cloud. You can also run over the falling missiles to destroy them.

Sevicog

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 PRINT"{CLR}HOW MANY TARGETS?{5 SPACES}(1-20)":INPUTN%:IFN%>
  20ORN%<1THEN1                                          :rem 28
2 GRAPHIC1:COLOR.,6,1,8:N%=N%-1:DIMM(N%,1),A(N%),B(N%):X=511:
  Y=X:SC=1                                              :rem 215
3 FORZ=.TON%:GOSUB900:NEXT:M=6.375:N=M*2:W%=1020:V%=960
                                                        :rem 61
5 DRAW2,.,W%TO54,987TO102,1011TO123,977TO198,1000TO300,1015TO
  333,978TO367,1001                                     :rem 68
```

```
6 DRAW2,367,1001TO400,975TO450,999TO511,969TO550,989TO575,101
  0TO634,999TO700,994                                    :rem 225
7 DRAW2,700,994TO724,1011TO800,971TO880,999TO902,970TO900,100
  9TO950,999TO1020,969                                    :rem 11
8 PAINT3,W%,W%                                            :rem 55
10 FORZ=.TON%:POINT.,M(Z,.),M(Z,1):M(Z,1)=M(Z,1)+M*2:M(Z,.)=(
   M(Z,1)-B(Z))/A(Z)                                     :rem 173
15 J=RJOY(.):GOSUB108:IFRJOY(.)>.THENGOSUB100            :rem 59
20 IFM(Z,.)>W%ORM(Z,.)<.THENGOSUB900:M(Z,1)=.            :rem 156
25 IFM(Z,1)>V%THEN970                                    :rem 211
30 IFRDOT(M(Z,.),M(Z,1))>2THENA=Z:GOSUB950:GOSUB900      :rem 113
35 IFQ%=.THEN50                                          :rem 108
40 FORA=.TON%:IFRDOT(M(A,.),M(A,1))>1THENM(A,1)=.:GOSUB950:GO
   SUB900:NEXT:GOTO50                                     :rem 66
45 NEXT                                                  :rem 168
50 POINT2,M(Z,.),M(Z,1):NEXT:CHAR.,.,S$:GOTO10           :rem 240
100 POINT.,X,Y                                            :rem 82
101 J=RJOY(.):X=X+(((JAND4)=4)-((JAND8)=8))*N*P:Y=Y+(((JAND1)
    =1)-((JAND2)=2))*M*P                                 :rem 239
102 IFJ>128THENP=P+1/(SC*.05):GOTO104                    :rem 136
103 P=1/(.05*SC):IFP>3THENP=3                            :rem 158
104 IFX<.THENX=.                                         :rem 216
105 IFY<.THENY=.                                         :rem 219
106 IFX>W%THENX=W%                                       :rem 120
107 IFY>V%THENY=V%                                       :rem 121
108 POINT3,X,Y:IFQ%=1ANDTI>SC*4THENI=I-1:GOSUB600:TI$="000000
    "                                                    :rem 246
109 IFJ<>128ORQ%<>.THENRETURN                             :rem 86
110 SOUND.,.,.,245,15:DRAW1,511,V%TOX,Y:DRAW.,511,V%TOX,Y:Y2=
    Y:GOSUB500                                             :rem 3
111 RETURN                                               :rem 115
500 Y2=ABS(Y2-20):FORI=1TO5:D(I)=ABS(X-INT(RND(.)*75)):DRAW3,
    D(I),Y2TOD(I)+65,Y2                                  :rem 174
505 SOUND.,.,.,200,15-I*3:Y2=Y2+M:NEXT:Q%=1:I=6:TI$="000000":
    SOUND.,.,.,.,.                                       :rem 175
510 RETURN                                               :rem 118
600 Y2=Y2-M:DRAW.,D(I),Y2TOD(I)+65,Y2:IFI=1THENQ%=.       :rem 24
610 RETURN                                               :rem 119
900 A(Z)=RND(.)*10-4.5:IFA(Z)=.THEN900                    :rem 92
905 M(Z,.)=INT(RND(Z)*700)+300:B(Z)=M(Z,1)-M(Z,.)*A(Z):RETURN
                                                         :rem 223
950 SOUND.,225,.,.,15:SC=SC+1:S$=STR$(SC-1):M(A,1)=SC*10:IFM(
    A,1)>800THENM(A,1)=800                               :rem 158
960 SOUND.,.,.,.,.:RETURN                                :rem 216
970 M(.,1)=M(Z,1):M(.,.)=M(Z,.):A(.)=50                  :rem 230
975 FORI=.TO40:B(.)=RND(.)*800+100:SOUND.,.,129,200,15:Q%=Q%+
    1:IFQ%=4THENQ%=1                                      :rem 69
980 DRAWQ%,M(.,.),M(.,1)TOB(.),A(.)                       :rem 60
985 DRAW.,M(.,.),M(.,1)TOB(.),A(.):SOUND.,.,.,I+128,15:NEXT
                                                         :rem 228
990 SOUND.,.,.,.,.:GRAPHIC4:PRINT"{CLR}SCORE: ";S$        :rem 51
```

Mike
Scharland

Pilot

You are serving as a gunner in the United States Air Force, and you're stationed at the fore of a squadron of backfire bombers. Your mission is to destroy enemy planes (a multicolor mode ball) which appear in front of the backfire bombers (after all, they can only shoot backward!). You will get a view from the cockpit window in which you can see the enemy jet in front of you weaving around in the air. You must maneuver your jet so that the enemy plane is centered in your view, to the place where the arrows at the side and top of the screen converge. To shoot at the jet, you press the joystick fire button.

But there is more to this than meets the eye. You must destroy the enemy jet before you run out of fuel and plunge to the earth. Also, a common trick of the enemy is to try to lure you into crashing into the mountain peaks, so you must try to maintain altitude. You'll also crash if your altitude is below 100 feet. You lose height when you pull down on the joystick and gain it when you push up. You have an altimeter at the bottom left which is updated every time you destroy an enemy jet.

How It Works

The play portion is a short loop which branches off to a number of subroutines. The moving of the enemy plane is done in lines 7 and 8. First, a random number is chosen and the data pointer is moved that far. The number that it's resting on is then added to the current position of the enemy fighter. Here's an outline of the subroutines:

1000–1012	Read joystick (see *Programmer's Reference Guide* for details)
2000–2009	Print game screen
3000–3003	Firing sequence
4000–4009	Explosion sequence and update score
5000–5007	End of game display
5999–6013	Screen for level input
7000–7200	New high-score sequence
8000–8006	Out of fuel sequence
9000–10002	Check current altitude

Some Hints

This game is difficult to get the hang of, and it may take several hours or even several days to get accustomed to the controls and get to the point where you can complete the game. You must always remember to keep complete control over the enemy plane by using the joystick. Always grip it firmly, because if you let go and the enemy jet floats off to the side of the

screen, it will take quite some time to get it back to the center. Also, if you see the message YOU'RE HEADED FOR A MOUNTAIN!! appear at the top of the screen, instantly push up on the joystick, and you'll be temporarily safe. The faster you destroy the plane in front of you, the better your score will be.

Good luck!

Pilot

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
0 PH=0:NAME$="NO ONE":G$="{32 SPACES}"                    :rem 218
1 GOSUB5999:POKE36879,184:TI$="000000":POKE36877,135:TU=0:AL=
  450:RESTORE                                             :rem 248
2 FORT=38444TO38839:POKET,8OR8:NEXT:GOSUB2000             :rem 82
5 T=TI:X=7910:POKEX,81                                    :rem 89
7 Z=INT(RND(1)*12)+1:FORM=1TOZ:READY:NEXT                 :rem 90
8 DATA 0,-23,-22,0,-21,-1,0,1,21,0,22,23                  :rem 100
9 GOSUB1000:POKEX,81                                      :rem 208
10 IF(TI-T)>5500THEN8000                                  :rem 94
11 RESTORE:GOTO7                                          :rem 48
1000 POKE37154,127:P=PEEK(37152)AND128:J0=-(P=0):POKE37154,25
     5:P=PEEK(37151)                                      :rem 6
1001 J1=-((PAND8)=0):J2=-((PAND16)=0):J3=-((PAND4)=0):FB=-((P
     AND32)=0)                                            :rem 20
1002 IFFBTHEN3000                                         :rem 204
1003 IFJ0THENY=Y+1:GOTO1010                               :rem 124
1004 IFJ2THENY=Y-1:GOTO1010                               :rem 129
1005 IFJ1THENY=Y+22:AL=AL-10:IFAL<100THEN10000:GOTO1010
                                                          :rem 20
1006 IFJ3THENY=Y-22:AL=AL+10                              :rem 159
1010 IFX+Y*O>8053ORX+Y*O<7724THENY=-Y                     :rem 5
1011 X=X+Y*O:POKEX-Y*O,32:POKEX,81                        :rem 84
1012 RETURN                                               :rem 164
2000 PRINT"{HOME}{7 DOWN}{BLU}I{LEFT}{DOWN}-{LEFT}{DOWN}-
     {LEFT}{DOWN}-{LEFT}{DOWN}JM{2 LEFT}{DOWN}UN{2 LEFT}
     {DOWN}-{LEFT}{DOWN}-{LEFT}{DOWN}-{LEFT}{DOWN}K"       :rem 144
2001 PRINT"{5 SPACES}{BLK}U*I{4 SPACES}U*I":PRINT"{5 SPACES}-
     {GRN}Q{BLK}-{4 SPACES}-{GRN}Q{BLK}-"                 :rem 93
2002 PRINT"{5 SPACES}-*J****K*-":PRINT"{RVS}£{4 SPACES}-****
     ****-{6 SPACES}{*}";                                 :rem 254
2003 PRINT"{RVS}{5 SPACES}J**I**U**K{2 SPACES}{3 J}{2 SPACES}
     ";:PRINT"{RVS}{5 SPACES}WW -**-{9 SPACES}";:GOTO9000
                                                          :rem 49
2004 POKE8185,160:POKE38905,0                             :rem 92
2008 PRINT"{HOME}{OFF}{BLU}{7 SPACES}J**I{LEFT}{DOWN}MN{LEFT}
     {UP}U***K"                                           :rem 201
2009 RETURN                                               :rem 171
3000 POKE36878,15:FORAZ=1TO30:NEXT:POKE36878,0:C=8108:FORA=7T
     O1STEP-1:POKEC+A,77                                  :rem 147
```

```
3001 POKEC-A,78:C=C-22:POKEC+22+A,32:POKEC+22-A,32:NEXT
                                                    :rem 184
3002 IFX=7954THEN4000                               :rem 181
3003 RETURN                                         :rem 166
4000 POKE36879,138:POKEX,32:POKEX-45,46:POKEX-22,46:POKEX-21,
     46:POKEX-2,46:POKEX-1,46                        :rem 69
4001 POKEX+1,46:POKEX+2,46:POKEX+22,46:POKEX+23,46  :rem 191
4003 FORI=15TO1STEP-2:POKE36878,I:FORV=1TO100:NEXT:NEXT:POKE3
     6878,0                                         :rem 215
4004 SC=SC+INT((T/TI)*321)+350*O:PRINT"{HOME}{PUR}{2 DOWN}SCO
     RE="SC;:PRINTG$                                :rem 157
4005 POKEX-45,32:POKEX-22,32:POKEX-21,32:POKEX-2,32:POKEX-1,3
     2:POKEX+1,32:POKEX+2,32                        :rem 250
4006 POKEX+22,32:POKEX+23,32:TU=TU+1:IFTU>7THEN5000  :rem 83
4007 POKE36879,184                                  :rem 212
4008 X=X-22-INT(RND(1)*5)+1:GOSUB2000                :rem 80
4009 T=TI:RETURN                                     :rem 21
5000 POKE36879,8:PRINT"{CLR}{2 DOWN}{CYN}OLD HIGH SCORE "PH:P
     RINT"{DOWN}BY: "NAME$                           :rem 17
5001 PRINT"{2 DOWN}{YEL}YOUR SCORE:"SC              :rem 242
5002 IFSC>PHTHEN7000                                :rem 184
5004 PRINT"{2 DOWN}{GRN}PRESS X TO PLAY AGAIN"      :rem 248
5005 GETA$:IFA$=""THEN5005                          :rem 183
5006 IFA$="X"THENSC=0:GOTO1                         :rem 110
5007 GOTO5005                                       :rem 207
5999 PRINT"{CLR}{BLU}":POKE36879,127                :rem 163
6001 PRINT"{3 DOWN}**ENTER SKILL LEVEL** {21 P}"     :rem 67
6005 PRINT"{2 DOWN}{2 SPACES}{RVS}1{OFF}{2 SPACES}EASY":PRINT
     "{2 DOWN}{5 SPACES}OR"                         :rem 147
6006 PRINT"{2 DOWN}{2 SPACES}{RVS}2{OFF}{2 SPACES}DIFFICULT"
                                                     :rem 47
6010 GETN$:IFN$=""THEN6010                          :rem 203
6011 O=VAL(N$):IFO<1ORO>2THEN5999                   :rem 238
6012 PRINT"{CLR}":POKE36865,255:GOSUB2000:FORI=140TO25STEP-1:
     POKE36865,I                                    :rem 176
6013 FORZS=0TO11:NEXT:NEXT:RETURN                   :rem 220
7000 PH=SC:POKE36877,0                                :rem 6
7001 PRINT"{DOWN}{RED}A ";:GOSUB7200:PRINT"NEW ";:GOSUB7200
                                                    :rem 120
7002 PRINT"HIGH ";:GOSUB7200:PRINT"SCORE!!":GOSUB7200:rem 196
7003 INPUT"{DOWN}{BLU}WHAT IS YOUR NAME";NAME$      :rem 142
7004 POKE36878,0:GOTO5004                           :rem 162
7200 POKE36878,15:POKE36876,200:FORE=1TO50:NEXT:POKE36876,0:R
     ETURN                                          :rem 17
8000 POKE36877,0:POKE36878,15:PRINT"{HOME}{3 DOWN}{RED}
     {2 SPACES}OUT OF FUEL!":RESTORE                :rem 18
8001 POKE36875,200:FORG=1TO300:NEXT:POKE36875,195:FORG=1TO300
     :NEXT:POKE36875,190                            :rem 74
8002 FORG=1TO300:NEXT:POKE36875,185:FORG=1TO300:NEXT:POKE3687
     5,0                                            :rem 172
8003 FORSA=25TO150:POKE36865,SA:NEXT                :rem 209
```

```
8004 POKE36879,42:POKE36877,135:FORWE=15TO0STEP-1:POKE36878,W
     E                                                  :rem 233
8005 FORET=1TO100:NEXT:NEXT                             :rem 225
8006 POKE36865,25:GOTO5004                              :rem 216
9000 PRINT"{HOME}{21 DOWN}{BLK}{RVS}"AL;"{OFF}{HOME}":GOTO200
     4                                                  :rem 158
10000 PRINT"{HOME}{2 DOWN}{BLK}YOU'RE HEADED FOR A{3 SPACES}M
      OUNTAIN!!":FORWQ=1TO400:NEXT                      :rem 153
10001 GOSUB1000:IFAL>=100THENPRINT"{2 UP}"G$:GOTO1010 :rem 87
10002 GOTO8004                                          :rem 248
```

Rob Westphal
and
Al Switzer

Laser War

"Laser War" is a simple two-player game for the unexpanded VIC. Each player attempts to shoot at the other player's ship. The ships shoot horizontally at each other. To move a ship one space vertically, press the appropriate up or down key once (each player's keys are listed on the instruction screen). To move more than one space vertically, press the appropriate key repeatedly. If there is a laser beam close to a ship, but not able to hit the ship directly, the ship will become invisible until the laser is gone. The first player to reach a score of five is the winner.

Laser War

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 POKE36879,27                                          :rem 54
15 PRINT"{CLR}":PRINTSPC(29):PRINT"{RED}L{CYN}A{PUR}S{GRN}E
   {BLU}R {YEL}W{RED}A{CYN}R"                            :rem 140
30 PRINTSPC(112):PRINT" (PRESS SPACE BAR)"               :rem 229
40 GETM$:IFM$=""THEN40                                    :rem 3
50 PRINT"{CLR}":PRINTSPC(27):PRINT"INSTRUCTIONS"         :rem 240
55 PRINT"{2 DOWN}{BLU}THIS IS DESIGNED FOR{7 SPACES}TWO PLAYE
   RS"                                                   :rem 148
60 PRINT"{DOWN}{RED}PLAYER 1{5 SPACES}{GRN}PLAYER 2"     :rem 127
65 PRINT"{RED}{DOWN}A=UP{9 SPACES}{GRN}K=UP"             :rem 215
70 PRINT"{RED}Z=DOWN{7 SPACES}{GRN}M=DOWN"                :rem 3
75 PRINT"{RED}X=FIRE{7 SPACES}{GRN}N=FIRE"               :rem 227
80 PRINT"{2 DOWN}{BLU}THE {PUR}X{BLU}'S IN THE MIDDLE OF THE
   {SPACE}SCREEN ARE{5 SPACES}BLOCKADES. SHOOTING"        :rem 62
85 PRINT"THEM WILL NOT HELP THESCORE OF THE GAME."       :rem 217
90 PRINT"{2 DOWN}{BLK}{2 SPACES}(PRESS SPACE BAR)"        :rem 11
100 GETM$:IFM$=""THEN100                                  :rem 93
295 CLR                                                  :rem 129
299 A=7834:B=8031                                        :rem 125
300 PRINT"{CLR}":POKE36879,8                             :rem 213
310 POKE36878,0:POKE36875,0                                :rem 1
320 POKEA,62:POKEB,60                                    :rem 214
400 FORT=38884TO38906:POKET,6:NEXT                        :rem 96
410 POKE8170,12:POKE8171,1:POKE8172,19:POKE8173,5:POKE8174,18
                                                         :rem 250
420 POKE8176,23:POKE8177,1:POKE8178,18                   :rem 159
430 POKE8168,27:POKE8180,29                                :rem 3
530 PRINT"{HOME}{BLU}SCORE=";SC;:PRINT"{3 SPACES}SCORE=";SK
                                                         :rem 253
531 T=7998:U=7866:V=8020:W=7844                          :rem 238
533 POKEV,86:POKE38740,4:POKEW,86:POKE38564,4            :rem 182
534 POKET,86:POKE38718,4:POKEU,86:POKE38586,4            :rem 188
545 GETA$                                                :rem 227
546 IFA$<>"A"THEN630                                      :rem 90
547 A=A-22                                               :rem 240
```

```
548 IFA=7680THENA=7702                                  :rem 0
549 POKEA+22,32:POKEA,62:GOTO630                        :rem 124
550 IFA$<>"Z"THEN650                                    :rem 112
555 A=A+22                                              :rem 237
557 IFA=8164THENA=8142                                  :rem 253
558 POKE36878,15:POKE36876,209                          :rem 177
559 POKE36878,0:POKE36876,0                             :rem 17
560 POKEA-22,32:POKEA,62:GOTO530                        :rem 118
570 IFA$<>"X"THEN530                                    :rem 109
573 POKE36878,10:POKE36875,209                          :rem 168
580 GOTO700                                             :rem 109
630 IFA$<>"K"THEN550                                    :rem 95
635 B=B-22                                              :rem 240
637 IFB=7701THENB=7723                                  :rem 254
640 POKEB+22,32:POKEB,60:GOTO530                        :rem 115
650 IFA$<>"M"THEN670                                    :rem 102
655 B=B+22                                              :rem 240
657 IFB=8185THENB=8163                                  :rem 6
658 POKE36878,15:POKE36874,209                          :rem 176
659 POKE36878,0:POKE36874,0                             :rem 16
660 POKEB-22,32:POKEB,60:GOTO530                        :rem 119
670 IFA$<>"N"THEN570                                    :rem 104
673 POKE36878,10:POKE36875,209                          :rem 169
680 GOTO800                                             :rem 111
700 C=A                                                 :rem 88
710 POKEC+1,67                                          :rem 255
715 Z=0                                                 :rem 100
720 C=C+1                                               :rem 184
725 Z=Z+1                                               :rem 235
730 IFC=BTHEN766                                        :rem 189
735 IFC=WTHEN300                                        :rem 199
736 IFC=UTHEN300                                        :rem 198
737 IFC=VTHEN300                                        :rem 200
738 IFC=TTHEN300                                        :rem 199
750 POKEC-2,32:POKEC,67                                 :rem 67
755 IFZ=21THEN757                                       :rem 252
756 GOTO720                                             :rem 116
757 POKE36878,0:POKE36875,0:GOTO300                     :rem 22
762 SK=SK+1:POKEA,104:POKED+2,32:POKED+1,32             :rem 20
764 POKE36879,13:GOTO770                                :rem 130
766 SC=SC+1:POKEB,104:POKEC-2,32:POKEC-1,32             :rem 11
?768 POKE36879,10:GOTO770                               :rem 131
770 POKE36878,0:POKE36875,0                             :rem 11
775 POKE36877,220                                       :rem 161
777 FORL=15TO0STEP-1                                    :rem 232
779 POKE36878,L                                         :rem 94
781 FORM=1TO75                                          :rem 81
783 NEXTM                                               :rem 46
785 NEXTL                                               :rem 47
787 POKE36877,0                                         :rem 64
789 POKE36878,0                                         :rem 67
```

```
790 IFSC=5THENPRINT"{CLR}{3 DOWN}{RED}":PRINTSPC(4):PRINT"PLA
    YER 1 WINS":PRINT"{BLU}":GOTO900                    :rem 203
792 IFSK=5THENPRINT"{CLR}{3 DOWN}{RED}":PRINTSPC(4):PRINT"
    {GRN}PLAYER 2 WINS":PRINT"{BLU}":GOTO900            :rem 244
798 GOTO299                                             :rem 133
800 D=B                                                  :rem 91
810 POKED-1,67                                            :rem 3
815 Y=0                                                 :rem 100
820 D=D-1                                               :rem 189
825 Y=Y+1                                               :rem 234
830 IFD=ATHEN762                                        :rem 186
835 IFD=WTHEN300                                        :rem 201
836 IFD=UTHEN300                                        :rem 200
837 IFD=VTHEN300                                        :rem 202
838 IFD=TTHEN300                                        :rem 201
850 POKED+2,32:POKED,67                                  :rem 68
855 IFY=21THEN857                                       :rem 253
856 GOTO820                                             :rem 118
857 POKE36878,0:POKE36875,0:GOTO300                      :rem 23
900 PRINT"{4 DOWN} DO YOU WANT TO PLAY{3 SPACES}AGAIN?
    {2 SPACES}(Y OR N)"                                 :rem 137
910 INPUTP$                                             :rem 158
920 IFP$="Y"THENRUN                                     :rem 156
922 POKE36879,27                                        :rem 114
925 PRINT"{CLR}"                                          :rem 4
930 END                                                 :rem 115
```

Andy
Wolber

Challenger One

This exciting action game for the unexpanded VIC lets you shoot enemies and increase the number of galaxies conquered. If you miss more than 15 percent of the enemies in your galaxy, you lose a life.

In this game you are given two lives to shoot as many enemy ships as possible. You control the sights with your joystick. The sights don't actually move, but rather the background moves. You have three seconds to kill the enemy; otherwise, a new one will appear.

Press the fire button to activate your lasers. Ten aliens will attack per galaxy (level). You can tell how many more you have to shoot by multiplying the level times ten, then subtracting the number of waves. A word of warning—this is a very difficult game to win. Learning to control the movement is difficult.

Challenger One

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 DEFFNA(B)=INT(RND(B)*415)+7725:POKE36879,10          :rem 83
20 FORM=0TO65:READN:POKE7168+M,N:NEXT                   :rem 95
30 DATA169,128,141,19,145,169,0,133,1,133,2,169,127,141,34,14
   5,162,119,236,32,145                                :rem 194
40 DATA208,4,169,1,133,1,169,255,141,34,145,162,118,236,17,14
   5,208,4,169,22,133,1                               :rem 190
50 DATA162,110,236,17,145,208,4,169,1,133,2,162,122,236,17,14
   5,208,4,169,22,133,2,96                             :rem 76
60 POKE36878,0:POKE36876,0:PRINT"{CLR}{YEL}{7 DOWN}{4 SPACES}
   CHALLENGER ONE"                                     :rem 62
70 PRINT"{2 DOWN}{4 SPACES}ANDREW{2 SPACES}WOLBER"    :rem 230
80 PRINT"{2 DOWN}{7 SPACES}LEVEL":INPUT"{10 LEFT}";B :rem 128
90 IFB>10ORB<1THENB=5                                  :rem 12
100 LE=B                                              :rem 161
110 SC=0:PK=0:LI=2                                     :rem 21
120 SH=0:PK=0                                         :rem 221
130 EP=(30-(LE*4)):EN=LE*10:NS=0:IFEP=<0THENEP=1       :rem 46
140 POKE36879,10:PRINT"{CLR}"                           :rem 0
150 FORA=7702TO7723:POKEA,102:POKEA+30720,7:NEXTA     :rem 152
160 FORB=8142TO8163:POKEB,102:POKEB+30720,7:NEXTB     :rem 155
170 FORC=7724TO8120STEP22:POKEC,102:POKEC+30720,7:NEXTC
                                                       :rem 62
180 FORD=7745TO8141STEP22:POKED,102:POKED+30720,7:NEXTD
                                                       :rem 73
190 WV=0                                              :rem 180
200 I=INT(RND(1)*416)+7725                            :rem 127
210 IFPEEK(I)<>32THEN200                              :rem 129
220 POKEI,42:POKEI+30720,3                            :rem 214
230 POKE36878,6:POKE36874,INT(RND(1)*75)+150          :rem 208
240 TI$="000000"                                      :rem 248
```

```
250 C1=INT(RND(1)*6)+1                                  :rem 166
260 WV=WV+1                                             :rem 139
270 GOSUB900:GOSUB800:C2=INT(RND(1)*6)+1                 :rem 78
280 PRINT"{HOME}{DOWN}{RIGHT}{RED}{RVS}GALAXY";LE:PRINT"
    {HOME}{DOWN}{14 RIGHT}{RED}{RVS}WAVE";WV            :rem 122
290 PRINT"{HOME}{21 DOWN}{RIGHT}{RED}{RVS}SCORE";SC     :rem 124
300 PRINT"{HOME}{RED}{21 DOWN}{13 RIGHT}{RVS}LIVES";LI
                                                        :rem 214
310 PRINT"{YEL}{HOME}{10 DOWN}[+]{7 SPACES}O{4 SPACES}P":PRIN
    T"{YEL}{HOME}{12 DOWN}[+]{7 SPACES}L{4 SPACES}@"    :rem 185
320 SYS7168                                             :rem 106
330 QP=INT(RND(1)*4)+1                                  :rem 208
340 IFQP=1THENDE=1:GOTO380                              :rem 105
350 IFQP=2THENDE=22:GOTO380                             :rem 158
360 IFQP=3THENDE=-1:GOTO380                             :rem 154
370 IFQP=4THENDE=-22:GOTO380                            :rem 207
380 IFPEEK(1)=1THENED=1:GOTO420                         :rem 110
390 IFPEEK(1)=22THENED=-22:GOTO420                        :rem 2
400 IFPEEK(2)=1THENED=-1:GOTO420                        :rem 149
410 IFPEEK(2)=22THENED=22                               :rem 198
420 ED=ED+DE                                            :rem 153
440 I=I+ED:IFPEEK(I)=102ORI=>8141ORI=<7724THENI=I-ED:GOTO470
                                                        :rem 159
450 POKEI-ED,32:POKEI-ED+30720,0                         :rem 67
460 POKEI+30720,C1:POKEI,42                              :rem 29
470 IFI=>7909ANDI=<7956ANDPEEK(37137)<95THEN490         :rem 251
480 GOTO540                                             :rem 110
490 IFI=>7909ANDI=<7912THEN530                          :rem 248
500 IFI=>7931ANDI=<7934THEN530                          :rem 239
510 IFI=>7953ANDI=<7956THEN530                          :rem 248
520 GOTO540                                             :rem 105
530 GOSUB710:GOSUB620:SC=SC+EP:SH=SH+1:POKEI+30720,3:FORD=1TO
    150:NEXT:POKEI,32                                   :rem 110
535 POKEI+30720,0                                       :rem 152
540 IFWV=ENTHEN560                                      :rem 111
550 GOTO600                                             :rem 105
560 PK=INT((SH*100)/WV):IFPK=>85THENLE=LE+1:GOSUB690:GOTO120
                                                        :rem 199
570 LI=LI-1:LE=LE:WV=0                                   :rem 78
580 IFLI=<0THENGOSUB660:GOTO60                          :rem 142
590 IFLI=>1THENGOSUB690:FORDI=1TO100:NEXT:GOTO120       :rem 139
600 IFTI$=>"000003"THENPOKEI,32:POKEI+30720,0:GOTO200  :rem 58
610 GOTO320                                             :rem 101
620 POKE36878,10                                        :rem 100
630 FORFD=254TO128STEP-4:POKE36876,FD:NEXT               :rem 99
640 POKE36878,0:POKE36876,0                               :rem 8
650 RETURN                                              :rem 123
660 PRINT"{HOME}{8 DOWN}{6 RIGHT}{YEL}GAME OVER"        :rem 170
680 FORTG=1TO500:POKE36878,10:POKE36876,200:NEXTTG:GOTO60
                                                        :rem 238
```

```
690 PRINT"{HOME}{9 DOWN}{7 RIGHT}{YEL}LEVEL";LE:FORG=1TO1000:
    NEXT                                                  :rem 124
700 PRINT"{CLR}":RETURN                                    :rem 21
710 FORZ=8121TO7953STEP-21                                 :rem 41
720 POKEZ+21,32:POKEZ+30720,3:POKEZ+30741,0:POKEZ,78:NEXTZ
                                                            :rem 4
730 POKEZ+21,32:POKEZ+30720,0:POKE8142,102:POKE8142+30720,7
                                                           :rem 65
740 FORJ=8140TO7956STEP-23                                 :rem 34
750 POKEJ+23,32:POKEJ+30720,3:POKEJ+30743,0:POKEJ,77:NEXTJ
                                                          :rem 186
760 POKEJ+23,32:POKEJ+30720,0                             :rem 109
770 POKE8163,102                                           :rem 94
780 POKE8163+30720,7                                       :rem 42
790 RETURN                                                :rem 128
800 S1=FNA(1):GH=S1:GOSUB870                              :rem 147
810 S2=FNA(1):GH=S2:GOSUB870                              :rem 150
820 S3=FNA(1):GH=S3:GOSUB870                              :rem 153
830 S4=FNA(1):GH=S4:GOSUB870                              :rem 156
840 S5=FNA(1):GH=S5:GOSUB870                              :rem 159
850 S6=FNA(1):GH=S6:GOSUB870                              :rem 162
860 S7=FNA(1):GH=S7                                        :rem 76
870 IFPEEK(GH)=102THENRETURN                               :rem 18
880 POKEGH,46:POKEGH+30720,C2                             :rem 180
890 RETURN                                                :rem 129
900 SA=30720                                              :rem 102
910 HG=S1:GOSUB980                                         :rem 69
920 HG=S2:GOSUB980                                         :rem 71
930 HG=S3:GOSUB980                                         :rem 73
940 HG=S4:GOSUB980                                         :rem 75
950 HG=S5:GOSUB980                                         :rem 77
960 HG=S6:GOSUB980                                         :rem 79
970 HG=S7                                                 :rem 246
980 IFPEEK(HG)=102THENRETURN                               :rem 20
990 POKEHG,32:POKEHG+SA,0:RETURN                           :rem 30
```

Chapter Nine

Adventures

Paul F.
McMillan

Space Mission

Earth has been engaged in a lengthy war with the aliens. You have been charged with patrolling a quadrant of space that contains 150 sectors. As you move from one sector to another, you may find either an empty sector, one which contains either a photon torpedo or a fuel module, one which contains one of the above objects and an alien warship, or one which contains only a warship.

When you first enter a new sector, you will have three options. The first is to move to another sector. The second is to take aboard the photon torpedo or fuel module, if any. The third is to engage the alien warship in battle. If you have encountered the alien for the third time since your last battle, you will have no choice but to fight. What object or objects, if any, are in a given sector is determined at runtime.

Entering the Program

Although "Space Mission" will run on an unexpanded VIC, it does require at least 8K expansion memory to enter. First, type in Program 1 with the expansion memory in place, and save it using the filename SPACE LOADER. Program 1 is a BASIC loader program that will write the machine language program to disk using the filename SPACE MISSION. Tape users should change the *,8,0* in line 50 to *,1,0.*

Once you have saved Program 1, enter the following line in direct mode:

POKE 44,30:POKE 7680,0:NEW

Now load Program 1 and run it. The program will take about a minute to POKE the machine language program into memory and save it to disk or tape. If you have made an error, the program will tell you. Correct the data and resave the program and run it again. If the program runs successfully, you will have the machine language on tape or disk under the filename SPACE MISSION.

From now on, to play Space Mission, simply load it into an *unexpanded* VIC using LOAD"SPACE MISSION",8,1 for disk, or LOAD"SPACE MISSION",1,1 for tape, and type SYS4558.

Program Operation

Upon initialization, you will be placed in sector 1 with a score of 5000. Your score is displayed on the first line at the extreme left. Your current sector number is the next number in the line, and the last number is the current number of alien warships. The number of alien warships is chosen randomly, but will be in the neighborhood of 75. The rest of the screen will be

filled with stars and any object(s) that may be in the sector. The star pattern will make an appropriate shift as you move about the quadrant.

Here is a complete list of the program commands and an explanation of how to use each one:

Movement is accomplished with keys 0–5. Keys 1–4 will move you forward from zero to four sectors. Just which key will do what depends on the sector you are in at the time. This is chosen at random on initialization, but remains constant for each sector for the run of the current game. Key 5 will move you backward from zero to four sectors. Again, just which it will do is chosen at random upon initialization for each sector and remains constant for each sector for the run of the current game. Backward movement from the first four sectors is never allowed. Pressing the 0 key will move you to a new sector chosen at random at that time.

To those of you who are new to adventure games, this is the time to mention the necessity of making a travel map to keep track of where your key choices will take you.

One last thing to mention on this subject is that forward movement beyond sector 150 will wrap around. For example, three forward from 149 will land you in sector 2.

Taking. Pressing the T key will take the photon torpedo or fuel module, if any, in the sector aboard your vessel and increase your score. An attempt to take a nonexistent object will result in an error message at the bottom of the screen, and the program will wait for valid input.

Fighting. The F key will engage you in battle with the alien warship. There are only two possible outcomes: Either you will destroy it or you will fail to do so. In this game you never get killed—unlike in most adventure games. Attempting to fight a nonexistent alien will result in an error message, and the program will wait for valid input.

Taking an object will remove it permanently from that sector for the rest of the game. The same goes for destroying an alien.

These are the only commands necessary to play the game, but here are two more that you might find convenient.

Quitting. If you are disgusted with your performance, you can reinitialize the program and start over by pressing the Q key. When you press Q, a prompt asks if you really want to do this just in case you pressed the key by mistake or want to change your mind.

Exiting. If you want to stop playing before the game is over and don't wish to resume it at a later time, you can exit the program and return to BASIC by pressing the X key. Once again, a prompt will ask you if you really mean it.

Before going on, let me remind you that if you encounter an alien warship for the third time since your last engagement, you must fight and there-

fore only the F key will work. A message at the bottom of the screen informs you of this.

Scoring

Space Mission uses the following scoring:

Moving to a new sector	−1
Fighting and winning	+100
Fighting and losing	−25
Taking a photon torpedo	+25
Taking a fuel module	+50

Strategy

The object of the game is to destroy all the alien warships while at the same time getting the highest score you can. When you fight or take an object, you are automatically moved to the next sector without the one-point penalty. After fighting, you can determine if you won or lost by noting the change in your score and/or the number of aliens. When all the aliens have been destroyed, the prompt line will ask you if you want a new game. An N response will exit you to BASIC.

In determining when to fight, you might want to consider that the outcome is determined by chance, but the odds will favor you when your score is above 5000 and favor the alien when below.

Space Mission

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 PRINT"{CLR}PLEASE WAIT...":I=4558                    :rem 212
20 READ A:IF A=256 THEN 40                               :rem 54
30 POKE I,A:I=I+1:C=C+A:GOTO 20                         :rem 235
40 IFC<>181780THENPRINT"TYPING ERROR IN DATA":STOP      :rem 127
50 SYS57809"0:SPACE MISSION",8,0:REM CHANGE TO ,1,0 FOR TAPE
                                                         :rem 158
60 POKE193,206:POKE194,17:POKE174,58:POKE175,25:SYS63109
                                                          :rem 4
100 DATA 160,154,56,165,1,101,4,101,5,133,0              :rem 32
110 DATA 162,4,181,0,149,1,202,16,249,201,0              :rem 39
120 DATA 144,234,201,5,176,230,153,189,25,136,208       :rem 104
130 DATA 224,234,234,234,234,234,234,234,234,234,234 :rem 248
140 DATA 234,234,234,234,234,234,169,8,141,15,144       :rem 106
150 DATA 169,147,32,210,255,169,136,141,35,25,169       :rem 119
160 DATA 19,141,36,25,169,1,141,37,25,169,0              :rem 65
170 DATA 141,38,25,141,39,25,160,150,56,165,1           :rem 160
180 DATA 101,4,101,5,133,0,162,4,181,0,149              :rem 244
190 DATA 1,202,16,249,201,167,176,45,201,140,144         :rem 48
```

```
200 DATA 4,162,1,176,39,201,115,144,7,162,2          :rem 49
210 DATA 238,39,25,176,28,201,25,144,7,162,3         :rem 115
220 DATA 238,39,25,176,17,201,12,144,7,162,4         :rem 111
230 DATA 238,39,25,176,6,162,5,144,2,162,0           :rem 13
240 DATA 138,153,39,25,136,208,180,169,147,32,210    :rem 113
250 DATA 255,162,15,169,31,133,2,189,40,24,24        :rem 163
260 DATA 109,37,25,201,228,208,2,169,227,133,1       :rem 211
270 DATA 169,46,145,1,202,208,234,162,15,198,2       :rem 218
280 DATA 189,40,24,24,109,37,25,133,1,169,46         :rem 123
290 DATA 145,1,202,208,240,169,25,72,169,28,72       :rem 223
300 DATA 169,0,72,173,37,25,72,32,232,22,173         :rem 110
310 DATA 28,25,133,0,169,10,133,1,133,3,32           :rem 246
320 DATA 146,23,169,25,72,169,28,72,169,0,72         :rem 131
330 DATA 173,39,25,72,32,232,22,173,28,25,133        :rem 162
340 DATA 0,169,16,133,1,133,3,32,146,23,169          :rem 55
350 DATA 25,72,169,28,72,173,36,25,72,173,35         :rem 132
360 DATA 25,72,32,232,22,173,28,25,133,0,169         :rem 110
370 DATA 1,133,1,133,3,32,146,23,172,37,25           :rem 255
380 DATA 185,39,25,201,0,240,49,201,1,208,6          :rem 56
390 DATA 32,191,23,76,58,19,201,2,208,9,32           :rem 20
400 DATA 191,23,32,225,23,76,58,19,201,3,208         :rem 109
410 DATA 6,32,225,23,76,58,19,201,4,208,9            :rem 222
420 DATA 32,225,23,32,6,24,76,58,19,32,6             :rem 171
430 DATA 24,173,39,25,201,0,208,3,76,61,21           :rem 4
440 DATA 173,38,25,201,3,208,23,162,11,169,162       :rem 208
450 DATA 133,0,169,24,133,1,32,174,23,32,228         :rem 103
460 DATA 255,201,70,208,249,76,9,22,32,228,255       :rem 226
470 DATA 162,83,208,3,76,23,20,162,76,208,3          :rem 69
480 DATA 76,129,20,201,81,208,3,76,239,20,201        :rem 162
490 DATA 88,208,3,76,23,21,201,84,208,3,76           :rem 26
500 DATA 89,21,201,70,208,3,76,9,22,201,48           :rem 11
510 DATA 208,33,56,165,1,101,4,101,5,133,0           :rem 244
520 DATA 162,4,181,0,149,1,202,16,249,201,0          :rem 44
530 DATA 144,234,201,151,176,230,205,37,25,208,74    :rem 102
540 DATA 240,174,201,49,208,2,240,45,201,50,208      :rem 250
550 DATA 2,240,39,201,51,208,2,240,33,201,52         :rem 91
560 DATA 208,2,240,27,201,53,240,2,208,144,174       :rem 202
570 DATA 37,25,224,6,176,2,144,135,189,189,25        :rem 186
580 DATA 133,0,138,56,229,0,76,252,19,56,233         :rem 125
590 DATA 48,24,109,37,25,170,189,189,25,24,109       :rem 240
600 DATA 37,25,201,151,144,2,233,150,141,37,25       :rem 195
610 DATA 173,35,25,56,233,1,141,35,25,173,36         :rem 112
620 DATA 25,233,0,141,36,25,76,111,18,234,234        :rem 154
630 DATA 234,234,162,12,169,173,133,0,169,24,133     :rem 56
640 DATA 1,32,174,23,32,228,255,201,89,208,2         :rem 112
650 DATA 240,17,201,78,208,243,162,11,169,32,157     :rem 61
660 DATA 228,31,202,16,250,76,98,19,169,147,32       :rem 233
670 DATA 210,255,169,27,141,15,144,174,37,25,189     :rem 75
680 DATA 39,25,201,2,208,2,240,10,201,3,208          :rem 43
690 DATA 2,240,4,201,4,208,3,206,38,25,169           :rem 11
700 DATA 1,162,1,160,255,32,186,255,169,0,32         :rem 108
```

```
710 DATA 189,255,169,35,133,0,169,25,133,1,162        :rem 223
720 DATA 88,160,26,169,0,32,216,255,0,162,12          :rem 112
730 DATA 169,185,133,0,169,24,133,1,32,174,23         :rem 165
740 DATA 32,228,255,201,89,208,2,240,17,201,78        :rem 218
750 DATA 208,243,162,11,169,32,157,228,31,202,16       :rem 57
760 DATA 250,76,98,19,169,147,32,210,255,169,27        :rem 39
770 DATA 141,15,144,169,1,162,1,160,255,32,186        :rem 215
780 DATA 255,169,0,32,189,255,169,0,162,255,160        :rem 28
790 DATA 255,32,213,255,169,8,141,15,144,76,111        :rem 18
800 DATA 18,234,234,234,234,234,234,234,234,234,234   :rem 205
810 DATA 234,234,234,234,234,234,234,234,234,234,234 :rem 254
820 DATA 234,234,234,234,234,234,234,234,234,162,12  :rem 201
830 DATA 169,197,133,0,169,24,133,1,32,174,23         :rem 169
840 DATA 32,228,255,201,78,208,13,162,12,169,32        :rem 12
850 DATA 157,228,31,202,16,250,76,98,19,201,89        :rem 234
860 DATA 208,232,76,206,17,162,12,169,209,133,0        :rem 12
870 DATA 169,24,133,1,32,174,23,32,228,255,201        :rem 213
880 DATA 78,208,13,162,12,169,32,157,228,31,202        :rem 17
890 DATA 16,250,76,98,19,201,89,208,232,0,162         :rem 183
900 DATA 16,169,221,133,0,169,24,133,1,32,174         :rem 157
910 DATA 23,32,228,255,201,78,208,1,0,201,89          :rem 111
920 DATA 208,244,76,206,17,172,37,25,185,39,25        :rem 234
930 DATA 201,1,208,23,169,0,32,223,21,238,37          :rem 103
940 DATA 25,173,37,25,201,151,208,5,169,1,141         :rem 162
950 DATA 37,25,76,111,18,201,2,208,26,169,3            :rem 68
960 DATA 32,223,21,238,37,25,173,37,25,201,151        :rem 212
970 DATA 208,5,169,1,141,37,25,234,234,234,76         :rem 177
980 DATA 111,18,201,4,208,24,169,3,32,244,21          :rem 107
990 DATA 238,37,25,173,37,25,201,151,208,5,169        :rem 231
1000 DATA 1,141,37,25,234,76,111,18,201,5,208         :rem 143
1010 DATA 23,169,0,32,244,21,238,37,25,173,37         :rem 158
1020 DATA 25,201,151,208,5,169,1,141,37,25,76         :rem 154
1030 DATA 111,18,162,20,169,237,133,0,169,24,133       :rem 45
1040 DATA 1,32,174,23,76,98,19,153,39,25,173          :rem 125
1050 DATA 35,25,24,105,25,141,35,25,173,36,25         :rem 154
1060 DATA 105,0,141,36,25,96,153,39,25,173,35         :rem 163
1070 DATA 25,24,105,50,141,35,25,173,36,25,105        :rem 200
1080 DATA 0,141,36,25,96,172,37,25,185,39,25          :rem 124
1090 DATA 201,0,240,10,201,1,240,6,201,5,240           :rem 68
1100 DATA 2,208,16,162,20,169,1,133,0,169,25           :rem 97
1110 DATA 133,1,32,174,23,76,98,19,56,165,1            :rem 67
1120 DATA 101,4,101,5,133,0,162,4,181,0,149            :rem 31
1130 DATA 1,202,16,249,201,0,176,2,144,232,201        :rem 184
1140 DATA 8,144,2,176,226,133,0,169,50,205,36         :rem 159
1150 DATA 25,240,4,16,17,48,23,169,0,205,35            :rem 57
1160 DATA 25,48,16,165,0,201,4,144,16,176,54          :rem 110
1170 DATA 165,0,201,5,144,8,176,46,165,0,201          :rem 103
1180 DATA 5,144,40,173,35,25,56,233,25,141,35         :rem 160
1190 DATA 25,173,36,25,233,0,141,36,25,169,0          :rem 109
1200 DATA 141,38,25,238,37,25,173,37,25,201,151          :rem 1
1210 DATA 208,5,169,1,141,37,25,76,111,18,173         :rem 161
```

```
1220 DATA 35,25,24,105,100,141,35,25,173,36,25        :rem 195
1230 DATA 105,0,141,36,25,185,39,25,201,2,208         :rem 148
1240 DATA 8,169,1,153,39,25,76,206,22,201,3            :rem 61
1250 DATA 208,8,169,0,153,39,25,76,206,22,169         :rem 177
1260 DATA 5,153,39,25,206,39,25,169,0,141,38          :rem 120
1270 DATA 25,238,37,25,173,37,25,201,151,208,5        :rem 214
1280 DATA 169,1,141,37,25,76,111,18,104,141,21        :rem 204
1290 DATA 25,104,141,22,25,104,141,23,25,104,141       :rem 29
1300 DATA 24,25,141,25,25,16,17,169,0,56,237          :rem 109
1310 DATA 23,25,141,23,25,169,0,237,24,25,141         :rem 148
1320 DATA 24,25,104,133,0,104,133,1,169,0,160         :rem 133
1330 DATA 0,145,0,169,0,141,26,25,141,27,25            :rem 46
1340 DATA 162,16,24,46,23,25,46,24,25,46,26            :rem 63
1350 DATA 25,46,27,25,56,173,26,25,233,10,168         :rem 169
1360 DATA 173,27,25,233,0,144,6,140,26,25,141         :rem 152
1370 DATA 27,25,202,208,221,46,23,25,46,24,25         :rem 157
1380 DATA 173,26,25,24,105,48,32,114,23,173,23        :rem 208
1390 DATA 25,13,24,25,208,187,173,25,25,16,5          :rem 118
1400 DATA 169,45,32,114,23,173,22,25,72,173,21        :rem 205
1410 DATA 25,72,96,72,160,0,177,0,168,240,9            :rem 68
1420 DATA 177,0,200,145,0,136,136,208,247,104,160      :rem 91
1430 DATA 1,145,0,160,0,177,0,24,105,1,145            :rem 242
1440 DATA 0,96,169,30,133,2,169,150,133,4,162         :rem 160
1450 DATA 1,160,0,189,28,25,145,1,169,5,145            :rem 65
1460 DATA 3,200,232,198,0,208,241,96,160,0,177        :rem 209
1470 DATA 0,153,228,31,169,5,153,228,151,200,202       :rem 46
1480 DATA 208,242,96,162,24,189,55,24,133,1,202        :rem 13
1490 DATA 189,55,24,133,0,169,91,160,0,145,0          :rem 116
1500 DATA 165,1,24,105,120,133,1,169,5,145,0           :rem 91
1510 DATA 202,208,225,96,162,42,189,119,24,133,1       :rem 57
1520 DATA 202,189,119,24,133,0,169,91,160,0,145         :rem 3
1530 DATA 0,165,1,24,105,120,133,1,169,5,145           :rem 94
1540 DATA 0,202,208,225,238,38,25,96,162,40,189        :rem 16
1550 DATA 79,24,133,1,202,189,79,24,133,0,169         :rem 174
1560 DATA 91,160,0,145,0,165,1,24,105,120,133         :rem 140
1570 DATA 1,169,5,145,0,202,208,225,96,1,12            :rem 57
1580 DATA 31,22,47,60,68,86,111,114,130,154,158        :rem 10
1590 DATA 176,182,211,186,30,189,30,209,30,210,30     :rem 105
1600 DATA 230,30,231,30,232,30,233,30,253,30,254       :rem 23
1610 DATA 30,18,31,21,31,171,30,192,30,193,30         :rem 140
1620 DATA 194,30,214,30,215,30,216,30,236,30,237       :rem 39
1630 DATA 30,238,30,2,31,3,31,4,31,24,31              :rem 142
1640 DATA 25,31,26,31,46,31,47,31,48,31,69             :rem 16
1650 DATA 31,136,30,138,30,158,30,160,30,180,30       :rem 245
1660 DATA 181,30,182,30,202,30,203,30,204,30,223       :rem 25
1670 DATA 30,224,30,225,30,226,30,227,30,245,30       :rem 243
1680 DATA 249,30,11,31,15,31,33,31,37,31,160           :rem 98
1690 DATA 13,21,19,20,32,6,9,7,8,20,160               :rem 113
1700 DATA 19,1,22,5,63,32,40,25,47,14,41              :rem 156
1710 DATA 160,12,15,1,4,63,32,40,25,47,14             :rem 201
1720 DATA 41,160,17,21,9,20,63,32,40,25,47              :rem 2
```

```
1730 DATA 14,41,160,5,24,9,20,63,32,40,25          :rem 205
1740 DATA 47,14,41,160,14,5,23,32,7,1,13           :rem 154
1750 DATA 5,63,32,40,25,47,14,41,160,14,15           :rem 4
1760 DATA 32,15,2,10,5,3,20,46,46,46,23            :rem 107
1770 DATA 1,9,20,9,14,7,160,14,15,32,1              :rem 58
1780 DATA 12,9,5,14,46,46,46,23,1,9,20              :rem 71
1790 DATA 9,14,7,32,0,0,0,0,0,0,0                   :rem 45
1800 DATA 0,0,0,0,0,0,0,0,0,0,0                    :rem 171
1810 DATA 0,0,0,0,0,0,0,0,0,0,0                    :rem 172
1820 DATA 0,0,0,0,0,0,0,0,256                       :rem 98
```

Richard Woods

Dungeon Escape

It's a mild spring day. The world is bursting with life and new beginnings. You've set out into the world to seek your fame and fortune, questing for gold, eager for battle with the forces of evil. But these are the latter times of the days of sorcery. The dragons are becoming extinct; the dungeons of legend have long been ransacked. Alchemy and adventure are being discarded for modern ways. It seems that adventurers like yourself have no place in this new world. The glorious legends are fading, crumbling like ancient parchment.

Yet 20 days ago, while passing through the town of Irstwile, you heard the talk in taverns of a Secret Place, recently sought by another traveler, a traveler never to return. You eagerly pressed for details, hoping that this is the way to your quest. The townsfolk couldn't tell you much or were unwilling to reveal what they really knew. By plying with drink and gold, you found some who told you of the ancient ruins of a once-great castle, occupied long ago by a savage ruler. They tell of the Dungeon, a vast underground labyrinth, constructed by the king to test the mettle of his Dark Knights. More, they would not say. The entire area within 20 miles of the Dungeon has been shunned for more than a century. Only the foolhardy dare to speak of it, let alone wish to find it. Unfortunately, nothing they could say would discourage you.

After more than a fortnight of travel, you come upon the jangled mass of stones, blackened and glossy as if scorched by unimaginable heat. Indeed, much time has passed since the evil castle once reached into the sky, for the area has been reclaimed by bush and wildflowers. It looks disappointingly harmless. As you clamber over the rocks, though, you notice a pile of fresh-cut rock, chipped and piled at the side of a dark, downward passage. Your excitement mounts, as the sky begins to roil with the towering thunderheads of a brewing storm. You descend into the darkness, sword in hand, and finally come upon a massive wooden door. You cannot find a way to open the door, and in your eagerness begin to flail at it with your sword. But this is no ordinary wood. Your sword breaks with the first blow, shattering into useless shards. Then the door opens, steaming with dank and horrible odors. Sword or no, you've come too far to turn back now. You enter the Dungeon, and the door slams behind you. Nothing you do will budge the giant door. You're trapped.

You light a torch and see before you a cryptic message:

In your world you find no rest;
Your blood is hot for glorious quest.
You're finished now, you foolish mortal
Without the keys to crack this portal.

Your mind is filled with fear as you realize what you've got yourself into. You then notice three small notches in the enormous door. If only you can find the keys to the door, you can escape to the safe world outside. This Dungeon is no place for gold and glory. It's a wicked trap that can be your crypt. Your only hope is to venture forth, down into the heart of the earth. With no sword, no food, and fading hope, you walk along the passageways, seeking the keys you need to escape the deadly Dungeon. And you haven't seen the worst yet. Some doors are locked, requiring you to bash at them, draining your precious, waning strength. If you can enter a room, you find it contains a chest. You never know if you should open a chest, for some are empty, while others spew forth noxious gases that weaken you further. But some contain food and drink to replenish you. Some contain a precious key. It's not going to be easy. Many rooms are guarded by fierce monsters, mutated creatures fashioned from dark, ancient magic. There is a sword somewhere within the Dungeon, but without it you don't dare fight these beasts. And fight you must, if you hope to find the keys. Good luck, adventurer. This quest of yours may be your doom.

Playing Dungeon Escape

It somehow seems less deadly from behind your TV screen. Move the ball representing your alter ego with a joystick plugged into the VIC controller port. The game requires no expansion memory; it runs on a 5K unexpanded VIC. After you run the game, the complete Dungeon appears. Move through the Dungeon and upward against a door to try to enter a room. Conserve your strength at all costs. Simply opening the door and entering a room costs you two strength points. You start out with a hundred points of strength. If the strength falls at or below zero, you surely know what happens to your desperate adventurer.

You cannot leave the Dungeon through the top of the screen until you've acquired at least three keys. Some doors are locked, but if you try repeatedly, you can open them, at a cost of five strength points. Chests filled with poisonous gases sap ten points of your strength. If you can find a chest with food and drink, six points are added to your strength. Somewhere, you'll find a sword with which you can face the monsters guarding some of the rooms. Fighting not only drains your energy, but it can also kill you, so you're allowed to retreat when you get too tired. You can press the S key at any time to get a status report of your remaining strength, whether or not you have the sword, the number of keys found, time elapsed, monsters killed, and number of times you've retreated from a fight or locked door. Don't use the status report after you've entered a room, or the chest will vanish. You can use the status indications to play competively with your

friends. Even if you don't find the keys, you can use time elapsed, monsters killed, and number of retreats to compare your ability.

Escaping from the Dungeon may seem impossible, but you do have just enough strength to find the keys. When you find all three keys, exit your persona through the top of the vertical passageway to complete the Dungeon. Again, we wish you luck. You'll need it.

Dungeon Escape

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
0 PRINT"{CLR}":DIMR(24):G=1:DEFFNB(B)=INT(RND(1)*10)+1
                                                    :rem 216
1 POKE36879,8:N=7801:S1=36874:S2=S1+1:S3=S1+2:S4=S1+3:V=S1+4:
  SP=100:TI$="000000"                                :rem 2
2 POKE36881,255:CA=0:W=160:D=121:C=7724:PRINT"{CLR}":GOSUB4:G
  OSUB9                                              :rem 31
3 C=38444:CA=0:W=2:D=6:GOSUB4:GOSUB9:GOTO10          :rem 139
4 FORC=CTOC+9:POKEC,W:NEXT:C=C+2:FORC=CTOC+9:POKEC,W:NEXT
                                                     :rem 0
5 FORI=1TO2:FORC=CTOC+21STEP3:POKEC,W:NEXT:C=C-2:NEXTI:rem 53
6 FORC=CTOC+21:POKEC,W:NEXT:C=C-20:FORC=CTOC+19:POKEC,D:C=C+2
  :NEXT                                              :rem 214
7 POKEC-12,32:C=C-1:POKEC,W:C=C+21:POKEC,W:C=C+1:CA=CA+1:IFCA
  =4THENRETURN                                       :rem 15
8 GOTO4                                              :rem 165
9 FORC=CTOC+22:POKEC,W:NEXT:RETURN                   :rem 26
10 POKE7734,120:POKE7735,120:POKE38454,2:POKE38455,2 :rem 193
11 FORI=1TOG:POKER(I),32:NEXT:POKE36881,24           :rem 78
12 GOSUB95:GETA$:IFA$="S"THENGOSUB25:GOTO2           :rem 141
13 POKE37154,127:A=(PEEK(37137)AND28)OR(PEEK(37152)AND128):A=
   ABS((A-100)/4)-7                                  :rem 236
14 ONAGOSUB16,17,15,,16,17,19,,,,18,17,16:POKE37154,255:GOTO1
   2                                                 :rem 132
15 F=N:N=N-1:GOTO20                                  :rem 125
16 F=N:N=N+22:GOTO20                                 :rem 175
17 F=N:N=N-22:GOTO20                                 :rem 178
18 F=N:N=N+1:GOTO20                                  :rem 126
19 RETURN                                            :rem 74
20 IFPEEK(N)=160THENN=F                              :rem 137
21 IFPEEK(N)=126THENGOSUB43                          :rem 162
22 IFPEEK(N)=120THENGOSUB50                          :rem 155
23 IFPEEK(N)=121THENGOSUB33:IFF<>NTHENGOSUB39        :rem 144
24 GOSUB82:RETURN                                    :rem 106
25 PRINT"{CLR}{5 SPACES}***STATUS***";SPC(49)"STRENGTH:"SP
                                                     :rem 26
26 IFSW=0THENPRINT"{DOWN}SWORD: NO"                  :rem 133
27 IFSW=1THENPRINT"{DOWN}SWORD: YES"                 :rem 219
28 PRINT"{DOWN}KEYS FOUND:"K;SPC(30)"TIME ELAPSED:"TI$;SPC(25
   )"MONSTERS KILLED:"MK                             :rem 233
29 PRINT"{DOWN}RETREATS:"RT:IFL=1THENRETURN          :rem 41
```

```
30 PRINT"{3 DOWN}HIT SPACE BAR TO GO ON"                    :rem 99
31 GETW$:IFW$=""THEN31                                      :rem 23
32 RETURN                                                   :rem 69
33 GOSUB80:SP=SP-2:GOSUB41:X=FNB(B):IFX>3THENGOTO53         :rem 35
34 PRINT"{HOME}LOCKED, TRY AGAIN:{RVS}FIRE{OFF}RETREAT:{RVS}R
   {OFF}"                                                   :rem 211
35 GOSUB92:IFB=0THEN33                                      :rem 96
36 RETURN                                                   :rem 73
37 PRINT"{HOME}{44 SPACES}":RETURN                          :rem 104
38 FORT=1TO2000:NEXT:RETURN                                 :rem 12
39 POKEN-45,126:RETURN                                      :rem 94
40 R(G)=N:G=G+1:GOSUB37:RETURN                              :rem 120
41 IFSP=0ORSP<0THEN75                                       :rem 79
42 RETURN                                                   :rem 70
43 P=INT(RND(1)*4)+1:TR$(1)="A KEY":TR$(2)="FOOD AND DRINK":T
   R$(3)="POISON GAS"                                       :rem 184
44 TR$(4)="A SWORD":GOSUB95:IFG>21ANDK<3THENP=1             :rem 91
45 IFP=1ANDK<3THENGOSUB91:GOSUB83:K=K+1                     :rem 234
46 IFP=2THEN:SP=SP+6:GOSUB91:GOSUB79:IFSP>100THENSP=100
                                                            :rem 46
47 IFP=3THEN:SP=SP-10:GOSUB91:GOSUB78:GOSUB41               :rem 160
48 IFP=4THEN:SW=SW+1:GOSUB85:IFSW>1THENSW=1:GOTO43          :rem 60
49 GOSUB38:GOSUB37:RETURN                                   :rem 150
50 IFK<3THENPRINT"{HOME}COWARD! YOU DON'T HAVE ALL OF THE KEY
   S"                                                       :rem 170
51 IFK<3THENN=F:GOSUB38:GOSUB37:RETURN                      :rem 18
52 IFK=3THENL=1:GOSUB25:PRINT"{DOWN}CONGRATULATIONS, YOU
   {3 SPACES}ESAPED!":GOTO76                                :rem 48
53 T=FNB(B):IFT>5THENGOSUB40:RETURN                         :rem 31
54 GOSUB95:PRINT"{HOME}A MONSTER IS GUARDING THE CHEST":GOSUB
   38:GOSUB37                                               :rem 108
55 PRINT"{HOME}FIGHT:{RVS}FIRE{OFF}{12 SPACES}RETREAT:{RVS}R
   {OFF}"                                                   :rem 11
56 GOSUB92:IFB=0THEN58                                      :rem 106
57 POKEN-22,121:RETURN                                      :rem 84
58 IFSW=1THENMS=0:GOTO61                                    :rem 42
59 GOSUB37:PRINT"{HOME}YOU WOULD DIE WITHOUT THE SWORD":GOSUB
   38                                                       :rem 153
60 GOSUB37:POKEN,121:N=F:RETURN                             :rem 236
61 M=FNB(B):H=FNB(B):IFM>7THEN67                            :rem 169
62 GOSUB37:PRINT"{HOME}YOUR MOVE":GOSUB38:GOSUB37           :rem 63
63 IFH<4THENPRINT"{HOME}MISSED":GOSUB89:GOSUB38:GOSUB37:GOTO6
   1                                                        :rem 214
64 PRINT"{HOME}YOU HIT THE MONSTER":MS=MS+1:GOSUB88:GOSUB38:G
   OSUB37                                                   :rem 191
65 IFMS=3THENPRINT"{HOME}YOU KILLED THE MONSTER":MK=MK+1:GOSU
   B38:GOSUB40:RETURN                                       :rem 56
66 GOTO61                                                   :rem 12
67 GOSUB37:PRINT"{HOME}MONSTER'S MOVE":GOSUB38:GOSUB37
                                                            :rem 151
```

```
68 IFH<5THENPRINT"{HOME}MISSED":GOSUB89:GOSUB38:GOSUB37:GOTO6
   1                                                  :rem 220
69 PRINT"{HOME}THE MONSTER HAS HIT{3 SPACES}YOU":SP=SP-12:GOS
   UB88:GOSUB38:GOSUB37                               :rem 218
70 GOSUB41                                             :rem 76
71 PRINT"{HOME}YOUR STRENGTH IS NOW:"SP:GOSUB38:GOSUB37
                                                      :rem 192
72 PRINT"{HOME}CONTINUE:{RVS}FIRE{OFF}{9 SPACES}RETREAT:{RVS}
   R{OFF}"                                            :rem 253
73 GOSUB92:IFB=0THEN61                                 :rem 99
74 POKEN-22,121:RETURN                                 :rem 83
75 L=1:GOSUB25:PRINT"{DOWN}YOU DIED":GOSUB88:GOTO77   :rem 129
76 FORA=8TO14:POKE36879,A:GOSUB83:NEXT:GOSUB79:GOTO77:rem 162
77 PRINT"{DOWN}PLAY AGAIN:{RVS}FIRE{OFF}":GOSUB92:RUN :rem 62
78 FORE=1TO15STEP.09:POKES4,251:POKEV,E:NEXT:FORE=15TO0STEP-.
   1:POKEV,E:NEXT:GOTO90                              :rem 107
79 FORL=225TO250STEP.3:POKEV,15:POKES1,L:NEXT:GOTO90  :rem 91
80 FORE=1TO3:POKES4,147:FORL=15TO0STEP-2:POKEV,L:FORM=1TO50:N
   EXTM,L:POKEV,0:NEXT                                :rem 251
81 GOTO90                                              :rem 11
82 POKES2,135:POKEV,5:FORT=1TO5:NEXT:GOTO90            :rem 19
83 POKES3,200:FORL=15TO0STEP-2:POKEV,L:FORM=1TO25:NEXTM,L:POK
   EV,0:GOTO90                                        :rem 174
84 RETURN                                              :rem 76
85 IFSW=1THENGOSUB91:POKEV,15:FORM=1TO3:POKES3,200:FORT=1TO10
   0:NEXT:POKES3,0                                    :rem 184
86 IFSW=1THENFORT=1TO30:NEXTT,M:POKES3,218:FORT=1TO550:NEXT:G
   OTO90                                              :rem 100
87 RETURN                                              :rem 79
88 FORM=235TO150STEP-1 :POKEV,15:POKES1,M:NEXT:GOTO90 :rem 90
89 POKES4,240:POKEV,15:FORE=15TO0STEP-.1:POKEV,E:NEXT:GOTO90
                                                      :rem 99
90 FORS=36874TO36878:POKES,0:NEXT:RETURN               :rem 74
91 PRINT"{HOME}THE CHEST CONTAINS{4 SPACES}"TR$(P):RETURN
                                                      :rem 138
92 B=PEEK(37137)AND32:IFB=0THENGOSUB37:RETURN          :rem 72
93 A$="":GETA$:IFA$="R"THENRT=RT+1:SP=SP-3:N=F:GOSUB37:GOSUB4
   1:RETURN                                           :rem 107
94 GOTO92                                              :rem 17
95 M=N+30720:POKEM,6:POKEN,81:IFF<>NTHENPOKEF,32:RETURN
                                                      :rem 247
96 RETURN                                              :rem 79
```

Chapter Ten

More Games

Bob
Appleton

Place Your Bet

You'll feel like you're at the races with "Place Your Bet" running on your unexpanded VIC. Place Your Bet is a race simulation for one to four players.

The game is self-explanatory and easy to play. Simply enter your bets when prompted. Keep in mind the odds and the amount of money you have remaining. There are ten races, and if you go bust, it can get pretty dull watching everyone else playing the odds.

Place Your Bet

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 V9=36879:V=36878:V5=36875:V6=36876:V7=36877:G=48:I%=30720:
   V1=7694                                                  :rem 49
20 GOSUB470                                                 :rem 125
30 B$(1)="{BLK}1-SPEEDY":B$(2)="{BLK}2-SANDY":B$(3)="{BLK}3-S
   COOTER":B$(4)="{BLK}4-SLEEPER"                           :rem 241
40 B$(5)="{BLK}5-SILLY":B$(6)="{BLK}6-SORRY":B$(7)="{BLK}7-SL
   IM"                                                      :rem 92
50 K%=7724:W%=0:S%=0:T%=0                                   :rem 72
70 GOSUB550                                                 :rem 129
75 FORX=1TO7:A%(X)=66*X-65:NEXTX                            :rem 208
80 PRINT"{CLR}":POKEV9,8                                    :rem 40
85 FORX=1TO7:POKE7679+X*66,48+X:POKE7679+I%+X*66,X:NEXTX
                                                            :rem 187
90 FORL=1TO21:POKE7700+L*22,89:POKE7700+I%+L*22,1:NEXTL
                                                            :rem 224
100 POKEV,15:FORM=254TO240+INT(RND(1)*10)STEP-1:POKEV6,M:NEXT
    M:POKEV6,0                                              :rem 158
110 X=INT(1+7*RND(TI))                                      :rem 242
120 IFX=W%ORX=S%THEN110                                     :rem 162
130 A%(X)=A%(X)+1                                           :rem 75
140 POKEK%+A%(X),81                                         :rem 3
150 POKEK%+A%(X)+1,75                                       :rem 99
160 POKEK%+A%(X)-1,74                                       :rem 101
170 POKEK%+A%(X)+1,73                                       :rem 99
180 POKEK%+A%(X)-1,85                                       :rem 105
190 POKEK%+I%+A%(X),X                                       :rem 144
200 POKEK%+I%+1+A%(X),X                                     :rem 228
210 POKEK%+I%-1+A%(X),X                                     :rem 231
220 POKEK%+I%-2+A%(X),0                                     :rem 193
230 IFA%(X)=X*66-47THENGOSUB780                             :rem 68
240 GOTO100                                                 :rem 96
250 POKEV9,29                                               :rem 236
260 PRINT"{CLR}{DOWN}{RED}{RVS}{8 SPACES}RESULTS{7 SPACES}
    {BLK}{OFF}"                                             :rem 143
270 PRINT"# CRAB"TAB(11)"5X PAYOFF"                         :rem 69
275 PRINT"[6 T]{5 SPACES}[9 T]"                             :rem 252
280 PRINTB$(W%);TAB(10);O%(W%)*5                            :rem 249
```

```
290 PRINT" ";B$(S%);TAB(11);O%(S%)*5/2                    :rem 211
300 PRINT"{2 SPACES}";B$(T%);TAB(12);O%(T%)*5/4           :rem 208
310 FORJ=1TOP%                                             :rem 75
320 IFC%(J)=W%THENS%(J)=S%(J)+B%(J)*O%(W%):GOTO360         :rem 22
330 IFC%(J)=S%THENS%(J)=S%(J)+B%(J)*O%(S%)/2:GOTO360      :rem 112
340 IFC%(J)=T%THENS%(J)=S%(J)+B%(J)*O%(T%)/4:GOTO360      :rem 117
350 S%(J)=S%(J)-B%(J)                                      :rem 42
360 NEXTJ                                                  :rem 34
370 PRINT"{DOWN}{BLU}{RVS}{8 SPACES}TOTALS{8 SPACES}{BLK}
    {OFF}";                                               :rem 225
380 FORJ=1TOP%                                             :rem 82
390 PRINT"{DOWN}";P$(J);TAB(10);S%(J);"X"                  :rem 49
398 IFC%(J)=W%ORC%(J)=S%ORC%(J)=T%THENGOTO900             :rem 163
399 GOSUB419                                              :rem 195
400 NEXTJ                                                  :rem 29
401 GOTO425                                               :rem 105
419 POKEV,15                                              :rem 181
420 FORT=220TO127STEP-1:POKE36874,T:POKEV5,T:FORM=1TO5:NEXTM:
    NEXTT:POKE36874,0:POKEV5,0                             :rem 74
421 RETURN                                                :rem 119
425 PRINT"{2 DOWN}{PUR}{RVS}PRESS ANY KEY{BLK}{OFF}";:rem 247
426 GETA$:IFA$=""THEN426                                   :rem 91
440 IFR%<>10THENPOKEV9,27:GOTO50                          :rem 211
450 PRINT"{CLR}":POKEV9,15                                :rem 135
451 PRINT"{HOME}{8 DOWN}{10 RIGHT}{RVS}{YEL}THE{2 DOWN}
    {3 LEFT}END{BLK}{OFF}":FORM=1TO500:NEXTM              :rem 133
455 T=506*RND(1):POKET+7680,88:POKET+38400,8*RND(1):GOTO455
                                                          :rem 159
470 PRINT"{CLR}{RVS}{RED} UQIPLACE YOUR BETUQI";          :rem 213
475 PRINT"{RED}{RVS}{23 SPACES}{OFF}{BLK}"                :rem 193
480 PRINT"{DOWN}{BLK}THERE ARE 7 CRABS{3 SPACES}":PRINT"IN EA
    CH RACE AND 10":PRINT"RACES EACH GAME."               :rem 212
490 PRINT"1-4 PLAYERS CAN PLAY."                           :rem 86
500 PRINT"EACH PLAYER STARTS":PRINT"WITH A STAKE OF 500X ":PR
    INT"AND CAN BET UP TO"                                :rem 250
510 PRINT"WHAT REMAINS IN THIS":PRINT"STAKE."              :rem 42
520 INPUT"{DOWN}{RIGHT}HOW MANY PLAYERS?{LEFT}";P%        :rem 104
530 IFP%<1ORP%>4THEN520                                    :rem 87
540 FORJ=1TOP%:INPUT"PLAYER'S NAME";P$(J):S%(J)=500:NEXTJ:RET
    URN                                                    :rem 12
550 DEFFNA(B)=INT(2+8*RND(TI))                            :rem 219
560 R%=R%+1                                                :rem 34
580 POKEV9,254:PRINT"{CLR}{RVS}RACE #";R%                 :rem 194
590 PRINT"{DOWN}# CRABS"TAB(11)"ODDS":PRINT               :rem 205
600 FORB=1TO7                                               :rem 7
610 O%(B)=FNA(B)                                           :rem 67
620 PRINT"";B$(B);TAB(10);O%(B);"TO 1"                    :rem 246
630 NEXTB                                                  :rem 26
640 PRINT"{22 T}"                                         :rem 109
650 FORJ=1TOP%                                             :rem 82
```

```
660 PRINTP$(J);"{2 SPACES}CRAB#";                          :rem 45
670 INPUT C%(J):IFC%(J)<1ORC%(J)>7THEN670                  :rem 78
680 PRINT;S%(J);:PRINT"LEFT BET";                           :rem 0
690 INPUT B%(J):IFB%(J)>S%(J)ORB%(J)<0THEN690              :rem 42
700 NEXTJ                                                  :rem 32
710 POKEV,15:READE%:IFE%=-1THENPOKEV,15:GOTO730           :rem 144
720 READD%:POKEV5,E%:FORT=1TOD%:NEXTT:POKEV5,0:GOTO710
                                                          :rem 191
730 POKEV7,135:FORL=15TO0STEP-.2:POKEV,L:NEXTL:POKEV7,0
                                                           :rem 21
740 FORT=1TO50:RESTORE:RETURN                             :rem 196
750 DATA215,80,225,80,232,80,235,80,235,40,235,40,235,80,232,
    80,232,40,232,40                                      :rem 239
760 DATA232,80,225,80,232,80,225,80,215,320,215,80,225,80,232
    ,80,235,80,235,40                                      :rem 40
770 DATA235,40,235,80,235,80,231,80,225,80,215,80,215,40,215,
    40,215,80,225,640,-1                                  :rem 178
780 IFW%=0THENW%=X:POKEV1,X+G:POKEV1+I%,X:RETURN          :rem 100
790 IFS%=0THENS%=X:POKEV1+2,X+G:POKEV1+I%+2,X:RETURN       :rem 23
795 IFT%=0THENT%=X:POKEV1+4,X+G:POKEV1+I%+4,X:FORT=1TO1500:NE
    XTT:GOTO250                                            :rem 43
900 POKEV,15:FORL=1TO15:FORM=250TO240STEP-1:POKEV6,M:NEXTM:FO
    RM=240TO250:POKEV6,M:NEXTM                            :rem 140
901 POKEV6,0:NEXTL:POKEV,0:GOTO400                        :rem 152
```

Bobby
Evangelista

Boogieball

"Boogieball" is a quick wild-goose chase for the standard VIC which requires a quick eye and fast reflexes. A Boogieball is an irritating ball that tries to boogie its way out of the arena. The only way to stop the Boogieballs from boogeying and escaping is to use your controllable Boogiepopper to pop the Boogieball before it reaches the walls of the arena. But wait, just as you pop the Boogieball, another one appears on the screen!

The Game

In other words....

Imagine yourself in a square arena somewhere on the strange planet of Astroyd. In the center of the arena is a small box from which Boogieballs exit. The Boogieballs are trying to conquer the outer surface of the planet.

Now the object of the game is to maneuver your Boogiepopper, using the assigned keys (I, J, K, and M), and to pop the randomly moving Boogieballs ascending from the box by simply running over them. If the Boogieballs dodge your moves and escape by passing through the arena walls, 10 points will be subtracted from your score and another Boogieball will appear on the screen.

Sounds easy? Not really. The game gets difficult as it progresses. A Boogieball may suddenly change into a Superboogieball worth 20 points as opposed to the 10 points for each Boogieball popped.

The Antagonist

No game is complete without an antagonist. Eventually, an evil Boogiepopper-stopper will ascend from the box and try to stop you, and you can't afford to be killed.

To make the game even more difficult, a Boogieball or a Superboogieball may suddenly change into one of these menacing Boogiepopper-stoppers. If you're anywhere around it, quick reflexes are demanded to make a quick get-away, unless, at the rate the Boogiepopper-stopper is moving, you can deliberately be killed by the stopper boogeying on top of you!

Typing the Program

When entering the program, don't use any spaces except in PRINT statements since the program will take up most of the memory. Also, note that lines 66 and 78 will exceed the standard limit of 88 characters per line number, so crunching is required by using the abbreviated keywords. For help on crunching a program, refer to Appendix D in your *Personal Computing Guide* which has a list of keyword abbreviations.

Good luck and may the popping force be with you!

Boogieball

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

Some lines of this program require keywords to be abbreviated so that they will not exceed the four-screen-line limit. See Appendix B.

```
1 HI=Ø:W=2ØØ:CC=3Ø72Ø:V=36878:GOTO67                          :rem 69
2 PRINTCHR$(147):SC=Ø:S1=V-2:Z1=122:Z2=76:Z3=8Ø:Z4=79:Y1=125:
  Y2=1Ø9:Y3=11Ø:Y4=112                                        :rem 228
3 P1=77Ø3:P2=77Ø4:P3=7725:P4=7726:C=32:A=7911:B=81            :rem 248
4 POKE8164,147:POKE8165,131:POKE8166,143:POKE8167,146:POKE816
  8,133:POKE8169,186                                          :rem 19
5 FORP=ØTO15:POKE817Ø+P,16Ø:POKE38891+P,1:NEXT                :rem 32
6 FORP=ØTO2:POKE7888+P,16Ø:POKE791Ø+P,16Ø:POKE7932+P,16Ø
                                                              :rem 41
7 POKE386Ø8+P,6:POKE3863Ø+P,6:POKE38652+P,6:NEXT              :rem 93
8 POKE38631,1:POKE7911,81                                     :rem 153
9 FORP=ØTO21:POKE768Ø+P,16Ø:POKE384ØØ+P,6:POKE8142+P,16Ø:POKE
  38862+P,6:NEXT                                              :rem 228
1Ø FORP=ØTO442STEP22:POKE768Ø+P,16Ø:POKE384ØØ+P,6:POKE77Ø1+P,
   16Ø:POKE38421+P,6:NEXT                                     :rem 218
11 REM                                                        :rem 7Ø
12 IFA=P1ORA=P2ORA=P3ORA=P4THEN48                             :rem 114
13 POKEP1,Z1:POKEP2,Z2:POKEP3,Z3:POKEP4,Z4                    :rem 186
14 POKEP1,Y1:POKEP2,Y2:POKEP3,Y3:POKEP4,Y4                    :rem 183
15 POKEP1,C:POKEP2,C:POKEP3,C:POKEP4,C                        :rem 15Ø
16 IFPEEK(197)=12THENP1=P1-22:P2=P2-22:P3=P3-22:P4=P4-22:GOSU
   B42                                                        :rem 246
17 IFPEEK(197)=44THENP1=P1+1:P2=P2+1:P3=P3+1:P4=P4+1:GOSUB43
                                                              :rem 41
18 IFPEEK(197)=36THENP1=P1+22:P2=P2+22:P3=P3+22:P4=P4+22:GOSU
   B44                                                        :rem 248
19 IFPEEK(197)=2ØTHENP1=P1-1:P2=P2-1:P3=P3-1:P4=P4-1:GOSUB45
                                                              :rem 47
2Ø POKEA,32                                                   :rem 99
21 X=INT(RND(1)*5)+1:IFX=1THENA=A-44                          :rem 1Ø3
22 IFX=2THENA=A+2                                             :rem 5
23 IFX=3THENA=A+44                                            :rem 61
24 IFX=4THENA=A-2                                             :rem 11
25 IFPEEK(A)<>32THENGOSUB47                                   :rem 166
26 POKEA,B                                                    :rem 7Ø
27 IFSC<ØTHEN65                                               :rem 148
28 IFSC>5ØTHENL=15Ø                                           :rem 128
29 IFSC<99THENPOKE8173,16Ø                                    :rem 5Ø
3Ø IFSC>1ØØTHENL=1ØØ                                          :rem 16Ø
31 IFSC<999THENPOKE8174,16Ø                                   :rem 1Ø1
32 IFSC>15ØTHENL=5Ø                                           :rem 123
33 IFSC>2ØØTHENL=25                                           :rem 122
34 IFSC>3ØØTHENL=1Ø                                           :rem 118
35 IFSC>4ØØTHENL=5                                            :rem 76
36 PRINTCHR$(19)"{22 DOWN}"CHR$(18)CHR$(5)SPC(6);SC;CHR$(145)
                                                              :rem 19
```

```
37 AA=INT(RND(1)*L)+1                                        :rem 157
38 IFAA=3THENB=42                                              :rem 0
39 IFAA=5THENB=87                                             :rem 12
40 POKEV,15:POKE36875,252:POKE36875,252:POKE36875,0          :rem 195
41 GOTO11                                                      :rem 0
42 IFPEEK(P1)=160ORPEEK(P2)=160THENP1=P1+22:P2=P2+22:P3=P3+22
   :P4=P4+22:RETURN                                          :rem 104
43 IFPEEK(P2)=160ORPEEK(P4)=160THENP1=P1-1:P2=P2-1:P3=P3-1:P4
   =P4-1:RETURN                                              :rem 168
44 IFPEEK(P3)ANDPEEK(P4)=160THENP1=P1-22:P2=P2-22:P3=P3-22:P4
   =P4-22:RETURN                                             :rem 212
45 IFPEEK(P1)=160ORPEEK(P3)=160THENP1=P1+1:P2=P2+1:P3=P3+1:P4
   =P4+1:RETURN                                              :rem 160
46 RETURN                                                      :rem 74
47 POKEV,15:POKE36877,250:FORT=1TO100:NEXT:POKE36877,0:A=7911
   :SC=SC-10:B=81:RETURN                                       :rem 24
48 POKEP1,77:POKEP2,78:POKEP3,78:POKEP4,77                     :rem 74
49 IFB=87THEN58                                              :rem 134
50 POKEV,15:POKES1,250:FORT=1TO100:NEXT:POKES1,0:POKEA,32:A=7
   911:SC=SC+10                                              :rem 249
51 IFB=42THENSC=SC+10:B=81                                     :rem 32
52 GOTO21                                                       :rem 3
53 PRINTCHR$(19)SPC(242)SPC(66)CHR$(29)" PLAY AGAIN?
   {2 SPACES}(Y/N)"                                            :rem 32
54 GETA$:IFA$=""THEN54                                       :rem 245
55 IFA$="Y"THEN2                                             :rem 153
56 IFA$="N"THENPOKE36879,27:PRINTCHR$(147)CHR$(31):END
                                                             :rem 174
57 GOTO54                                                      :rem 14
58 FORPP=15TO1STEP-.5                                          :rem 55
59 POKEP1,42:POKEP2,42:POKEP3,42:POKEP4,42                     :rem 42
60 POKEP1,86:POKEP2,86:POKEP3,86:POKEP4,86                     :rem 66
61 POKEP1,77:POKEP2,78:POKEP3,78:POKEP4,77                     :rem 69
62 POKEP1,C:POKEP2,C:POKEP3,C:POKEP4,C                       :rem 152
63 POKE36877,150:POKEV,PP                                    :rem 244
64 NEXT:POKE36877,0:B=81                                     :rem 159
65 IFSC>HITHENHI=SC                                          :rem 242
66 PRINTCHR$(19)SPC(110)CHR$(18)CHR$(5)SPC(7)"GAME OVER":PRIN
   TCHR$(17)SPC(7)CHR$(18)CHR$(5)"HI:";HI:GOTO53             :rem 175
67 POKE36879,8:PRINTCHR$(147)SPC(116)CHR$(30)"BOOGIEBALL"CHR$
   (5)                                                       :rem 229
68 FORT=1TO3000:NEXT                                         :rem 246
69 PRINTCHR$(147)CHR$(17)"THE OBJECT OF THE GAME{4 SPACES}IS
   {SPACE}TO POP THE"                                          :rem 31
70 PRINT"{4 SPACES}BOOGIEBALL BY"                            :rem 163
71 PRINT"MANEUVERING YOUR '+'."                                :rem 32
72 PRINTCHR$(17)CHR$(159)"{3 SPACES}↑"                         :rem 67
73 PRINTCHR$(17)"{3 SPACES}I"                                  :rem 62
74 PRINT"< J K >{18 SPACES}M"SPC(44)CHR$(157)"V"             :rem 192
```

```
75 PRINTCHR$(19)SPC(163)CHR$(5)"Q "CHR$(158)"GET THIS,"
                                                    :rem 154
76 PRINTCHR$(17)SPC(9)CHR$(5)"* "CHR$(158)"AND THIS,":rem 132
77 PRINTSPC(9)CHR$(5)"W "CHR$(158)"BUT NOT"SPC(16)"THIS!"
                                                     :rem 88
78 PRINTCHR$(17)CHR$(3Ø)"POINTS ARE SUBTRACTED{3 SPACES}IF TH
   E BOOGIEBALL{4 SPACES}REACHES THE BORDER."        :rem 93
79 PRINTCHR$(17)CHR$(5)" Q=1Ø PTS. *=2Ø PTS.{5 SPACES}W=BLOWS
    YOU UP!"                                         :rem 98
8Ø PRINTCHR$(17)CHR$(18)CHR$(159)"HIT ANY KEY TO START"
                                                    :rem 134
81 GETA$:IFA$=""THEN81                              :rem 245
82 GOTO2                                            :rem 213
```

Robert
Flatman

Light Cycles

In this two-player game, which requires the *Super Expander* cartridge, lines of light race across the screen, each trying to outmaneuver the other. The game itself is quite simple. Cut off your opponent without hitting anything—walls, your own trail, or your opponent's track. If you're not careful, you can even get caught in an exploding light cycle.

Type in "Light Cycles," save it, and (making sure you have the *Super Expander* cartridge plugged in) load and run it. The game screen immediately appears.

The red player controls the light cycle with a joystick. The blue player, however, uses the keyboard. Pressing four keys moves the blue cycle in the following directions:

P Up
L Left
. Down
: Right

As your cycle moves across the screen, notice that it picks up speed. The farther it moves in one direction, the faster it travels. This lets you race your opponent to a wall as you try to cut him or her off.

Don't backtrack—your light cycle will explode!

Once you've trapped your opponent, a score is displayed, based on the length of your trail and the time you survived. You have the options of playing a new game, restarting the previous one (retaining the point totals so far), or quitting.

Light Cycles

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
Ø REM SET KEY REPEAT,SET VARIABLES TO STARTING POSITIONS
                                                        :rem 5Ø
5 POKE65Ø,128:BF=2:FF=1:GOSUB1ØØ:RS%=Ø:BS%=Ø            :rem 72
1Ø SCNCLR:COLOR1,6,Ø,8:GR%=RGR(Ø):IFGR%<>1THEN:GRAPHIC1
                                                       :rem 157
15 REM CLEAR SCREEN:{2 SPACES}SET COLOR:GRAPHIC MODE :rem 115
2Ø DRAW2,6Ø,6ØTO966,6ØTO966,966TO6Ø,966TO6Ø,6Ø         :rem 78
3Ø DRAW1,Ø,ØTO1Ø23,ØTO1Ø23,1Ø23TOØ,1Ø23TOØ,Ø          :rem 142
35 REM DRAW BORDER                                      :rem 56
4Ø GOTO2ØØ:REM GAME START                              :rem 245
1ØØ DX%=1:DY%=Ø:RX=1Ø7.4:RY=1Ø7.4:XR%=1Ø7:YR%=1Ø7:SR=18Ø:R%=8
                                                       :rem 248
1Ø5 R%=RJOY(Ø)                                          :rem 15
11Ø SY%=DY%:SX%=DX%                                    :rem 1Ø6
12Ø XD%=-1:YD%=Ø:BX=924:BY=924.6:XB%=924:YB%=924.6:BR=147:G$=
    "L"                                                 :rem 69
```

```
130 YS%=YD%:XS%=XD%                                        :rem 108
150 RETURN:REM SEE LINE 0                                  :rem 201
200 FORF2=1TOINT(RND(1)*30)+10:FORF1=1TOFF:REM HOW LONG AND H
    OW MANY MOVES RED GETS                                 :rem 188
205 R%=RJOY(0)                                              :rem 16
210 REM R%=JOYSTICK INPUT AND WHERE TO GO                  :rem 178
220 ON R%GOTO250,260,0,270,230,230,0,280                     :rem 1
230 GOTO300                                                 :rem 97
250 DY%=-1:DX%=0:GOTO290:REM DY%ANDDX% ARE DIRECTIONS RED MOV
    ES                                                      :rem 15
260 DY%=1:DX%=0:GOTO290                                     :rem 62
270 DY%=0:DX%=-1:GOTO290                                   :rem 108
280 DY%=0:DX%=1:GOTO290                                     :rem 64
290 IF(DY%<>SY%)AND(DX%<>SX%)=-1THEN:SOUND0,0,0,0,15        :rem 26
292 REM BREAK SOUND ON DIRECTION CHANGE                     :rem 83
295 SY%=DY%:SX%=DX%:SR=180                                  :rem 45
300 RX=RX+DX%*12.8:RY=RY+DY%*6.4:XR%=RX:YR%=RY:SR=SR+0.4
                                                           :rem 223
305 REM VARIABLES FOR{2 SPACES}MOVEMENT                    :rem 103
310 SOUND0,SR,BR,230,2:REM SR=RED SOUND:BR=BLUE            :rem 148
320 RD%=RDOT(XR%,YR%):REM CHECK TO SEE IF RED HIT SOMETHING
                                                           :rem 219
330 IFRD%<>1 THEN 800                                       :rem 82
340 POINT3,XR%,YR%                                          :rem 75
350 NEXT                                                   :rem 215
400 FORF3=1TOBF:REM{3 SPACES}BLUE MOVEMENTS                :rem 145
405 G=PEEK(197):REM{2 SPACES}KEYBOARD INPUT AND WHERE TO GO
                                                           :rem 186
410 IFG=64THENGOTO500                                       :rem 15
420 ONINT(G/10)GOTO480,470,450,460                         :rem 109
430 GOTO500                                                :rem 101
450 YD%=1:XD%=0:GOTO490                                     :rem 65
460 YD%=0:XD%=1:GOTO490                                     :rem 66
470 YD%=0:XD%=-1:GOTO490                                   :rem 112
480 YD%=-1:XD%=0:GOTO490                                   :rem 113
490 IF(YD%<>YS%)AND(XD%<>XS%)=-1THEN:SOUND0,0,0,0,15:REM SEE
    {SPACE}292                                             :rem 180
495 YS%=YD%:XS%=XD%:BR=147                                  :rem 33
500 BX=BX+XD%*12.8:BY=BY+YD%*6.4:XB%=BX:YB%=BY:BR=BR+0.4
                                                            :rem 63
505 REM SEE 305                                            :rem 243
510 SOUND0,SR,BR,230,2                                     :rem 255
520 BD%=RDOT(XB%,YB%):REM CHECK IF BLUE HIT SOMETHING:rem 122
530 IFBD%<>1 THEN 900                                       :rem 69
540 POINT1,XB%,YB%                                          :rem 43
550 REM                                                    :rem 126
560 NEXT                                                   :rem 218
600 CC%=CC%+1:NEXT:BF=3-BF:FF=3-FF:GOTO20:REM REVERSE SPEED A
    DVANTAGE                                               :rem 223
800 GOSUB1300:REM EXPLODE RED AND CHECK BLUE                :rem 63
```

```
810 IFRDOT(BX+XD%*12.8,BY+YD%*6.4)<>1THENGOSUB1400:GOSUB1500:
    GOSUB100:GOSUB950                                    :rem 227
815 GOSUB1500                                            :rem 228
820 DX%=1:DY%=0:RX=107.4:RY=107.4:XR%=107:YR%=107:SR=180:R%=8
                                                          :rem 1
830 SY%=DY%:SX%=DX%                                      :rem 115
840 BS%=BS%+CC%:CC%=0                                    :rem 117
850 GOTO950                                              :rem 116
860 REM ADD SCORE TO RED                                  :rem 69
900 GOSUB1400:REM SAME AS 800-,EXCEPT FOR BLUE           :rem 127
910 IFRDOT(RX+DX%*12.8,RY+DY%*6.4)<>1THENGOSUB1300:GOSUB1500:
    GOSUB100:GOSUB950                                     :rem 3
915 GOSUB1500                                            :rem 229
920 RS%=RS%+CC%:CC%=0                                    :rem 148
930 XD%=-1:YD%=0:BX=924:BY=924.6:XB%=924:YB%=924.6:BR=147:G$=
    "L"                                                   :rem 78
940 YS%=YD%:XS%=XD%                                      :rem 117
950 GRAPHIC4                                             :rem 208
955 REM PRINT SCORE AND OPTIONS                          :rem 143
960 PRINT"{3 DOWN}{6 RIGHT}{RED}RED {BLK}";RS%           :rem 221
970 PRINT"{3 DOWN}{6 RIGHT}{BLU}BLUE{BLK}";BS%            :rem 30
972 DI%=RS%-BS%                                           :rem 66
974 IFBS%>RS%THENDI%=BS%-RS%                             :rem 196
976 PRINT"{3 DOWN}{3 RIGHT}DIFFERENCE";DI%               :rem 185
980 PRINT"{3 DOWN}{5 SPACES}1:NEW GAME"                   :rem 20
990 PRINT"{5 SPACES}2:CONTINUE WITH{9 SPACES}GAME"       :rem 154
1000 PRINT"{5 SPACES}3:QUIT GAME":POKE198,0              :rem 195
1100 GETE$:IFE$=""THEN1100                               :rem 175
1110 ONVAL(E$)GOTO5,10,2000                              :rem 230
1200 GOTO1100                                            :rem 190
1300 FORI=1TO10                                          :rem 102
1305 REM DRAW RANDOM RED LINES                           :rem 242
1310 DRAW3,RX,RYTORX+INT(RND(1)*100)-50,RY+INT(RND(1)*100)-50
                                                          :rem 43
1320 NEXT                                                  :rem 5
1330 RETURN                                              :rem 167
1400 FORI=1TO10                                          :rem 103
1405 REM DRAW RANDOM BLUE LINES                           :rem 64
1410 DRAW1,BX,BYTOBX+INT(RND(1)*100)-50,BY+INT(RND(1)*100)-50
                                                         :rem 234
1420 NEXT                                                  :rem 6
1430 RETURN                                              :rem 168
1500 FORI=254TO135STEP-1:REM MAKE EXPLOSION SOUND         :rem 42
1510 SOUNDI,0,0,0,15                                      :rem 63
1520 NEXT                                                  :rem 7
1530 SOUND0,0,0,0,0:RETURN                                :rem 12
2000 END                                                 :rem 153
```

E. L.
Hayno

Rescue Mission

You need an unexpanded VIC, a joystick, and quick reflexes to complete "Rescue Mission."

You have just flown into a courtyard, and your job is to get as many people onto open ground as possible. Bullets are flying from the gun tower, people are popping in and out, and the courtyard walls are high. You must remain observant of the people popping out in the open. Rescue them only by sweeping down and picking them up; they're the only ones who will earn you points. Each time you happen to get shot or crash, you'll lose three points and one of your four turns. Time is short (about a minute and a half) and the people are panicking, so work fast. Flying high puts you out of range of gunfire and allows you to fly faster.

Typing In the Program

Note that periods are used in place of zeros to speed up the program. In order to get the program to fit into the unexpanded VIC, certain memory-saving techniques were necessary. Be sure not to add any extra spaces.

Rescue Mission

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 PRINT"{CLR}":POKE36879,29:PRINT"{6 DOWN}{4 RIGHT}{RVS}RESCU
  E MISSION"                                          :rem 138
6 POKE52,29:POKE56,29:CLR:A=25600:TI$="000000"         :rem 58
7 FORI=7176TO7431:POKEI,PEEK(I+A):NEXT                :rem 109
8 FORI=7504TO7631:POKEI,PEEK(I+A):NEXT                :rem 107
9 POKE37139,.:B=37154:C=37137:D=37152                   :rem 2
10 E=30720:F=36878:G=8142:H=8164:W=8126:J=7697:K=.:Q=8:U=8080
                                                      :rem 148
12 READA:IFA=-1THEN22                                 :rem 248
13 FORN=ATOA+7:READM:POKEN,M:NEXT:GOTO12              :rem 127
14 DATA7384,255,16,112,143,142,112,204,204,7392,16,16,112,143
   ,142,112,204,204                                   :rem 237
15 DATA7400,24,36,56,255,16,40,40,108,7408,153,90,60,255,24,6
   0,90,153                                           :rem 122
17 DATA7416,16,16,16,56,56,124,124,124,7432,255,255,255,255,2
   4,24,24,24                                         :rem 232
19 DATA7440,225,162,230,140,152,156,248,252,7448,254,255,255,
   6,198,70,70,254                                    :rem 241
20 DATA7456,128,96,50,25,14,11,31,31,7464,127,255,255,96,99,9
   8,98,127,-1                                         :rem 49
22 PRINT"{CLR}{UP}{2 RIGHT}{PUR}RESCUE MISSION{BLU}":GOSUB180
   :POKEF-9,255:Z=TI                                  :rem 213
23 FORT=.TO3:POKEJ+T,27:POKEJ+E+T,.:NEXT               :rem 29
24 X=INT(RND(1)*20):Y=INT(RND(1)*15):IFX<.ORX>21ORY<.ORY>21TH
   ENX=10:Y=10                                        :rem 165
```

```
26 POKESC,32:SC=7702+X+22*Y                                  :rem 48
27 POKESC,27:POKESC+E,2:POKEF-1,200:FORL=9TO15STEP.7:POKEF,L:
   NEXT                                                     :rem 136
28 POKEF,4:POKEF-1,180:POKESC,28                            :rem 166
30 POKEB,127:O%=(PEEK(C)AND28)OR(PEEK(D)AND128)             :rem 230
32 P%=ABS((O%-100)/4)-7                                     :rem 136
35 ONP%GOSUB80,81,82,,83,84,87,,,,88,89,90                  :rem 213
36 GOTO26                                                    :rem 10
80 Y=Y+1:X=X-1:GOSUB94:RETURN                               :rem 121
81 X=X-1:Y=Y-1:GOSUB94:RETURN                               :rem 124
82 X=X-1:GOSUB94:RETURN                                     :rem 246
83 Y=Y+1:GOSUB94:RETURN                                     :rem 247
84 Y=Y-1:GOSUB94:RETURN                                     :rem 250
87 GOSUB94:RETURN                                           :rem 118
88 X=X+1:GOSUB94:RETURN                                     :rem 250
89 X=X+1:Y=Y-1:GOSUB94:RETURN                               :rem 130
90 X=X+1:Y=Y+1:GOSUB94:RETURN                               :rem 120
94 IFX<.THENGOSUB160                                          :rem 4
95 IFX>21THENGOSUB160                                        :rem 60
96 IFY>21THENY=21                                            :rem 32
97 IFY<.THENY=.                                             :rem 181
100 IFABS(TI-Z)>15*360THENR=4:POKEF,.:GOTO166               :rem 221
103 IFSC<8054THENRETURN                                     :rem 213
104 IFPEEK(8079)=28THEN150                                  :rem 222
106 IFPEEK(8054)=28THEN150                                  :rem 217
108 IFPEEK(8106)=28THEN155                                  :rem 222
110 IFPEEK(8107)=28THEN155                                  :rem 216
112 IFSC<8120THENRETURN                                     :rem 207
114 IFPEEK(G+1)=28THENGOSUB160                               :rem 43
115 IFPEEK(H+1)=28THENGOSUB160                               :rem 45
117 IFPEEK(G+2)=28THENGOSUB160                               :rem 47
119 IFPEEK(H+2)=28THENGOSUB160                               :rem 50
120 IFPEEK(H+11)=28THENGOSUB160                              :rem 90
122 IFPEEK(G+11)=28THENGOSUB160                              :rem 91
124 IFPEEK(H+14)=28THENGOSUB160                              :rem 97
126 IFPEEK(G+14)=28THENGOSUB160                              :rem 98
127 IFPEEK(W)=28THENGOSUB160                                :rem 227
129 IFPEEK(W+1)=28THENGOSUB160                               :rem 65
130 IFPEEK(W+22)=28THENGOSUB160                             :rem 108
132 IFPEEK(W+23)=28THENGOSUB160                             :rem 111
134 IFPEEK(W+44)=28THENGOSUB160                             :rem 116
136 IFPEEK(W+45)=28THENGOSUB160                             :rem 119
138 IFPEEK(H+K)=28THENS=S+10:GOTO173                        :rem 178
139 IFPEEK(H+Q)=28THENS=S+1:GOTO173                         :rem 137
141 RETURN                                                  :rem 118
150 POKEU,46:POKEU+E,.:POKEU+23,46:POKEU+23+E,.              :rem 52
152 FORV=.TO2:FORL=14TO.STEP-2:POKEF,L:FORT=1TO20:NEXT:NEXT:N
    EXT                                                      :rem 10
154 POKEU,32:POKEU+23,32:GOTO160                            :rem 152
155 POKE8106,46:POKE8106+E,.:FORL=14TO.STEP-2:POKEF,L:FORT=1T
    O20:NEXT:NEXT                                           :rem 173
```

```
156 POKE8106,32                                          :rem 43
160 POKEF+1,31:S=S-3                                     :rem 117
161 POKESC,30:POKESC+E,4:POKEF-1,195:FORL=15TO.STEP-1:POKEF,L
    :FORT=1TO100:NEXT:NEXT                               :rem 62
164 POKEJ,32:J=J+1:R=R+1:POKEF+1,29                      :rem 37
166 IFR=4THENPOKEF+1,8:PRINT"{CLR}{WHT}{10 DOWN}{6 RIGHT}*GAM
    E OVER*"                                             :rem 248
167 IFR=4THENPRINT"{2 DOWN}SCORE ";S                     :rem 28
170 IFR=4THENPRINT"{DOWN}HIT FIRE TO PLAY AGAIN{BLK}":WAIT371
    37,32:WAIT37137,32,32:GOTO5                          :rem 18
171 POKESC,32:GOSUB175:GOTO30                            :rem 28
173 POKEF-1,.:POKEF,15:POKEF-2,220:FORT=1TO10:NEXT:POKEF-2,.
                                                         :rem 159
175 PRINT"{HOME}{BLK}{3 DOWN}SCORE{4 SPACES}{3 LEFT}";S:FORT=
    1TO1200:NEXT                                         :rem 231
176 POKESC,32:X=INT(RND(1)*20)+.:Y=INT(RND(1)*15):SC=7702+X+2
    2*Y:POKESC,28                                        :rem 83
177 Q=INT(RND(1)*21):K=INT(RND(1)*20):IFK=QTHENK=Q-1     :rem 84
178 IFQ=1ORQ=2ORQ=6ORQ=7ORQ=11ORQ=14THENQ=4              :rem 48
180 POKESC,32:POKEG+1,31:POKEG+1+E,5:POKEG+2,31:POKEG+2+E,5:P
    OKEH+1,33:POKEH+1+E,5                                :rem 96
182 POKEH+2,33:POKEH+2+E,5:POKEG+11,31:POKEG+11+E,5:POKEH+11,
    33:POKEH+11+E,5                                      :rem 154
185 POKEG+14,31:POKEG+14+E,5:POKEH+14,33:POKEH+14+E,5    :rem 42
190 POKEW+1,34:POKEW+1+E,2:POKEW,36:POKEW+E,2:POKEW+22,35:POK
    EW+22+E,.                                            :rem 216
192 POKEW+23,37:POKEW+23+E,.:POKEW+44,35:POKEW+44+E,.:POKEW+4
    5,37                                                 :rem 80
193 POKEW+45+E,.                                         :rem 129
194 POKEH+Q,29:POKEH+Q+E,6:POKEH+K,29:POKEH+K+E,.:RETURN
                                                         :rem 240
200 END                                                  :rem 105
```

Robert L. Lykins

Scavenger Hunt

"Scavenger Hunt," for the unexpanded VIC, is a perfect game for a Halloween party. But play it once and you may be hooked. It's colorful and has plenty of action and excitement. As your play improves, you reach more difficult levels but are rewarded with higher point values.

The object of the game is to collect—in only five minutes—10 each of the following items: clocks, jack-o'-lanterns, steins, televisions, and umbrellas. To reach the next level, you must also collect at least 25 bonus items: candles, hot dogs, and lamps. All of these items appear at the bottom of the screen and move upward. You use the joystick to move a little figure around the screen to collect the objects. You may stop the scrolling by pressing the fire button.

Sounds easy, right? There are complications. Scattered all over the screen are hedges. If you hit too many of them in a short period of time, you lose all your bonus items and are delayed for about five seconds. Stopping the scrolling with the fire button helps you gather objects and avoid hedges, but not without risk. It also releases a ghost from one of the four corners. The ghost's intent is to get you. Releasing the button stops the ghost dead. Pressing it again will release another ghost. The result of contacting a ghost is devastating. You lose all items collected in that round.

If you obtain your quota and collect enough bonus items, you will advance to the next level. In the second round, play is the same, but you must run into fewer hedges. Ghosts will rush you twice as fast.

It's quite difficult to reach the third level, but not impossible. In the third round, you may not run into even one hedge without penalty. Ghosts move at triple speed. Can you make it to level four?

Typing In the Programs

The programs as listed here are written for disk. If you use a Datassette, delete line 337, and replace lines 330 and 335 with the following:

```
330 PRINT"{CLR}{3 DOWN}PLEASE WAIT FOR PART 2{DOWN}TO
    LOAD.{DOWN}{WHT}
335 POKE198,1:POKE631,131:END
```

Before beginning to enter the program, remove all expansion memory. Then type in Program 1 and save it, after making sure there are no errors. Next, enter and save Program 2. Tape users should be sure to save Program 2 on the same tape immediately following Program 1. Disk users must save Program 2 with the filename SCAVENGER.2.

Now, reload Program 1. When you run the program, it will take about six seconds to load the machine language and character information. It will

then ask if you need instructions. If you do, press the joystick fire button. When you finish with the instructions, or if you didn't need them, Program 2 will automatically load. You may then center the screen with the cursor keys and press the fire button to start play.

Program 1. Scavenger Hunt

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
100 REM SCAVENGER HUNT{11 SPACES}PART ONE                :rem 107
105 IFPEEK(56)<>30THENPRINT"{CLR}NO EXPANSION, PLEASE.":END
                                                         :rem 237
110 POKE36879,136                                        :rem 152
115 PRINT"{CLR}{4 DOWN}{4 RIGHT}{RVS}{WHT}SCAVENGER HUNT
                                                         :rem 133
120 PRINT"{3 DOWN}{2 RIGHT}{RVS}{BLK}REQUIRES{2 SPACES}JOYSTI
    CK                                                    :rem 49
125 POKE52,28:POKE56,28:FORC=7552TO7631:POKEC,PEEK(C+25600+12
    8*8):NEXT                                            :rem 112
130 FORL=1TO11:READA:READB:FORC=0TO7:POKE7168+A*8+C,PEEK(3276
    8+B*8+C):NEXT:NEXT                                   :rem 106
135 FORL=1TO11:READA:FORC=0TO7:READB:POKE7168+A*8+C,B:NEXT:NE
    XT                                                   :rem 244
140 FORL=7168TO7359:READA:POKEL,A:NEXT                   :rem 136
145 DATA32,32,62,158,61,159,63,42                        :rem 141
150 DATA44,147,45,148,46,143,47,144,58,140,59,129,60,153
                                                          :rem 7
155 DATA35,62,127,107,127,127,127,127,85,34,42,186,162,234,16
    8,174,138,171                                        :rem 134
160 DATA36,28,42,127,119,93,34,28,0,37,28,62,127,8,8,8,8,12
                                                         :rem 145
165 DATA38,30,126,94,94,94,126,30,63,39,62,65,81,73,73,73,62,
    0                                                     :rem 51
170 DATA41,48,48,168,168,168,168,168,168,42,0,255,255,170,170
    ,255,255,0                                           :rem 251
175 DATA43,252,252,252,252,252,32,32,168,40,34,20,8,127,69,71
    ,69,127                                               :rem 79
180 DATA33,28,8,28,62,93,28,20,54                        :rem 142
185 DATA169,8,141,15,144,169,255,141                      :rem 43
190 DATA5,144,169,240,141,14,144,169                      :rem 32
195 DATA127,141,34,145,169,10,141,254                     :rem 80
200 DATA28,141,255,28,96,169,4,45                        :rem 146
205 DATA17,145,208,10,169,0,205,255                      :rem 229
210 DATA28,240,3,206,255,28,169,8                        :rem 138
215 DATA45,17,145,208,10,169,21,205                      :rem 230
220 DATA255,28,240,3,238,255,28,169                      :rem 244
225 DATA16,45,17,145,208,10,169,0                        :rem 132
230 DATA205,254,28,240,3,206,254,28                      :rem 229
235 DATA169,119,205,32,145,208,10,169                     :rem 84
240 DATA20,205,254,28,240,3,238,254                      :rem 227
245 DATA28,169,240,205,14,144,240,3                      :rem 231
```

```
250 DATA206,14,144,96,173,253,28,201                    :rem 28
255 DATA0,208,5,170,168,76,146,28                       :rem 146
260 DATA201,1,208,7,162,20,160,21                       :rem 112
265 DATA76,146,28,201,2,208,7,162                       :rem 140
270 DATA0,160,21,76,146,28,162,20                       :rem 126
275 DATA160,0,142,251,28,140,252,28                     :rem 228
280 DATA96,173,251,28,205,254,28,240                    :rem 38
285 DATA11,48,6,206,251,28,76,172                       :rem 146
290 DATA28,238,251,28,173,252,28,205                    :rem 39
295 DATA255,28,240,11,48,6,206,252                      :rem 191
300 DATA28,76,191,28,238,252,28,96                      :rem 204
305 PRINT"{3 DOWN}{WHT}{2 SPACES}NEED INSTRUCTIONS?     :rem 143
310 PRINT"{DOWN}PRESS FIRE BUTTON NOW.{BLK}             :rem 149
315 FORL=1TO500:IFNOTPEEK(37137)AND32THEN340            :rem 165
320 NEXT                                                :rem 212
325 POKE36879,15:POKE36869,240                          :rem 167
330 PRINT"{CLR}{YEL}PLEASE WAIT {SHIFT-SPACE}           :rem 5
335 POKE198,2:POKE631,13:PRINT"{HOME}{BLK}":PRINT"{2 DOWN}LOA
    D";CHR$(34);"SCAVENGER.2";CHR$(34);",8"             :rem 160
337 POKE632,13:PRINT"{5 DOWN}RUN":PRINT"{HOME}":END     :rem 123
340 POKE36879,60:POKE36869,242                          :rem 166
345 PRINT"{CLR}{BLK}":FORL=1TO9:READA$:IFA$="←"THEN325
                                                        :rem 217
350 PRINTA$:NEXT:PRINT"{3 DOWN}{BLU} {RVS}(PRESS{SHIFT-SPACE}
    FIRE{SHIFT-SPACE}BUTTON)";                          :rem 77
355 IFPEEK(37137)AND32THEN355                           :rem 171
360 FORL=1TO99:NEXT:GOTO345                             :rem 215
365 DATA"YOU HAVE FIVE MINUTES{DOWN}","TO COLLECT 10 EACH OF
    {DOWN}","THE FOLLOWING ITEMS:{DOWN}                 :rem 219
370 DATA"CLOCKS,JACK-O-LANTERNS","STEINS, TELEVISIONS,{DOWN}"
    ,"AND UMBRELLAS. FOR{DOWN}                          :rem 126
375 DATA"BONUS POINTS, YOU MAY{DOWN}","COLLECT AS MANY OF
    {DOWN}","THESE AS YOU WISH:                         :rem 52
380 DATA"CANDLES,HOTDOGS,LAMPS.","YOU USE THE JOYSTICK{DOWN}"
    ,"TO MANEUVER AROUND{DOWN}                          :rem 120
385 DATA"HEDGES AND PICK UP THE","OBJECTS. PRESS THE{DOWN}","
    FIRE BUTTON TO STOP{DOWN}                           :rem 31
390 DATA"THE SCREEN MOVEMENT{DOWN}","FOR EXTRA CONTROL, BUT",
    "BEWARE THE GHOST WHICH                             :rem 7
395 DATA"WILL RUSH YOU FROM ANY","CORNER. RELEASING THE{DOWN}
    ","BUTTON WILL KILL THE{DOWN}                       :rem 133
400 DATA"GHOST, BUT HE IS STILL","DANGEROUS.{DOWN}","{DOWN}
                                                        :rem 245
405 DATA"YOU WILL NOT RECEIVE{DOWN}","BONUS POINTS UNLESS
    {DOWN}","YOU REACH YOUR QUOTA.                      :rem 37
410 DATA"YOU LOSE YOUR BONUS{DOWN}","ITEMS IF YOU TOO OFTEN",
    "COLLIDE WITH HEDGES.{DOWN}                         :rem 235
415 DATA"YOU LOSE ALL ITEMS IF{DOWN}","YOU MAKE CONTACT WITH
    {DOWN}","A GHOST. TO REACH THE{DOWN}                :rem 65
420 DATA"NEXT LEVEL, YOU MUST{DOWN}","OBTAIN YOUR QUOTA AND
    {DOWN}","COLLECT 25 BONUSES.                        :rem 32
425 DATA"←                                              :rem 54
```

Program 2. Scavenger.2

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
100 REM SCAVENGER HUNT{11 SPACES}PART TWO                :rem 131
105 R=RND(-TI):POKE36879,10:PRINT"{CLR}{4 DOWN}{4 RIGHT}{GRN}
    {RVS}SCAVENGER HUNT                                  :rem 108
110 PRINT"{4 DOWN}{CYN}{2 SPACES}CENTER SCREEN WITH{7 SPACES}
    {DOWN}CURSOR KEYS                                     :rem 12
115 PRINT"{4 DOWN}{YEL}{RIGHT}{RVS}PRESS{2 SPACES}F B
    {2 SPACES}TO BEGIN                                   :rem 116
120 GETA$                                                :rem 216
125 IFA$="{RIGHT}"THENPOKE36864,PEEK(36864)+1:IFPEEK(36864)=9
    THENPOKE36864,2                                       :rem 35
130 IFA$="{DOWN}"THENPOKE36865,PEEK(36865)+1:IFPEEK(36865)=35
    THENPOKE36865,16                                     :rem 123
135 IFPEEK(37137)AND32THEN120                            :rem 157
140 PRINT"{CLR}                                          :rem 215
145 SYS7168:DEFFNR(R)=RND(1)*R:SG=7421:DEFFNP(P)=7680+X+Y*22:
    DEFFNG(G)=7680+XG+YG*22                              :rem 229
150 V=36878:CO=36879:BZ=36875:H=10:U=36876:LV=1:D$="{HOME}
    {7 DOWN}                                              :rem 66
155 FORM=0TO2STEP2:FORL=1TO22:READA:NEXT:POKEV,250       :rem 243
160 FORL=1TO7:READA:READB:POKEBZ,A:POKEU-M,A:FORP=0TO99*B:NEX
    T:NEXT:POKEV,240                                      :rem 42
165 RESTORE:POKEU-M,0:NEXT                               :rem 205
170 C=30720:PX=7422:PY=7423:DEFFNJ(J)=NOTPEEK(37137)ANDJ:POKE
    U,240                                                :rem 186
175 K=2:TI$="000000                                      :rem 209
180 GOSUB535                                             :rem 182
185 O=FNR(25):IFO>9THENO=0                                :rem 66
190 Z=0:POKEFNP(P),32                                    :rem 224
195 ON(O+1)GOSUB480,485,490,495,500,505,510,515,520      :rem 139
200 GOSUB525                                             :rem 174
205 SYS7197                                              :rem 110
210 X=PEEK(PX):Y=PEEK(PY)                                 :rem 53
215 IFZ=1ANDPZ<>FNP(P)THENPOKEPZ,32                        :rem 4
220 PZ=FNP(P)                                              :rem 0
222 IFPEEK(PZ)>33THENON44-PEEK(PZ)GOSUB350,350,350,365,365,36
    5,365,365,375,410                                    :rem 136
225 POKEC+PZ,7:POKEPZ,33                                 :rem 232
230 IFTI$>"000500"THEN285                                 :rem 90
235 IFH<10THENH=H+1                                      :rem 102
240 IFFNJ(32)THENGOSUB445:Z=1:GOTO205                     :rem 17
245 GOTO185                                              :rem 114
250 BO=BO+Q:Q=0:BN=0:FORP=1TO5:Q(P)=0:NEXT               :rem 106
255 LV=LV+1                                              :rem 121
260 FORP=1TO9:FORL=1TO9:POKEV,240+L:POKEBZ,200+P*2+L*4:POKECO
    ,P*2+20*L:NEXT:NEXT                                  :rem 255
265 POKEV,240:POKECO,8                                    :rem 67
270 PRINT"{CLR}":PRINTTAB(119)LV                          :rem 41
275 FORL=1TO2000:NEXT                                     :rem 29
```

```
280 GOTO175                                                    :rem 112
285 GOSUB530:GOSUB545                                           :rem 15
290 POKEV,240                                                  :rem 226
295 FORL=1TO3000:NEXT                                           :rem 32
300 IFQ<7500*LVTHEN325                                          :rem 16
305 FORP=1TO9:FORL=1TO9:POKEV,240+L:POKEBZ,200+P*2+L*4:POKECO
    ,P*2+20*L:NEXT:NEXT                                        :rem 255
310 POKEV,240:POKECO,8                                          :rem 58
315 FORL=1TO2000:NEXT                                           :rem 24
320 GOTO250                                                    :rem 101
325 PRINT"{DOWN}{3 RIGHT}{PUR}= ,-./{4 RIGHT}{GRN}> /:;<
                                                               :rem 240
330 IFFNJ(4)THENRUN140                                          :rem 65
335 IFFNJ(16)THENPRINT"{CLR}{BLU}";:POKE36869,240:POKE37154,2
    55:POKECO,27:END                                           :rem 190
340 L=L+.01:IFL>1THENL=0:POKECO,FNR(255)                       :rem 140
345 GOTO330                                                    :rem 107
350 BN=BN+1:POKEBZ,250                                          :rem 25
355 POKEV,247:RETURN                                             :rem 5
360 POKEBZ,235:POKECO,8:GOTO355                                :rem 153
365 IFQ(41-PEEK(PZ))<>10THENPOKECO,15:Q(41-PEEK(PZ))=Q(41-PEE
    K(PZ))+1:GOTO360                                           :rem 101
370 RETURN                                                     :rem 122
375 POKEBZ,0:POKECO,25:POKEV,250                               :rem 214
380 FORL=1TO11:READA:READB:POKEU,A:FORM=0TO99*B:NEXT:POKEU,0:
    NEXT                                                            :rem 3
385 RESTORE:POKEV,240:POKEU,240:POKECO,8                        :rem 36
390 GOSUB530:FORL=0TO9:GOSUB545:NEXT                            :rem 59
395 FORL=1TO163:PRINTD$SPC(L)" {WHT}#":FORP=1TO20:NEXT:NEXT:B
    N=0:FORP=1TO5:Q(P)=0:NEXT                                  :rem 115
400 FORL=0TO9:GOSUB545:NEXT:GOSUB535                            :rem 56
405 SYS7276:RETURN                                             :rem 136
410 H=H-5*LV:IFH<0THEN420                                      :rem 210
415 POKECO,138:FORL=1TO9:POKEBZ,175-L*5:POKEV,253-L:NEXT:POKE
    CO,8:RETURN                                                :rem 112
420 FORM=1TO6:GOSUB415:NEXT:H=10:POKEV,240                     :rem 127
425 GOSUB530                                                   :rem 179
430 FORL=1TO20:GOSUB545:NEXT                                    :rem 14
435 GOSUB535                                                   :rem 185
440 BN=0:RETURN                                                :rem 175
445 IFZ=0THENPOKESG,FNR(4):SYS7276:GOTO460                     :rem 158
450 POKEGE,OP:POKEC+GE,CP                                       :rem 65
455 FORL=1TOLV:SYS7321:NEXT                                      :rem 3
460 XG=PEEK(7419):YG=PEEK(7420):GE=FNG(G):OP=PEEK(GE):CP=PEEK
    (C+GE)                                                      :rem 35
465 POKEC+GE,1:POKEGE,35                                       :rem 174
470 IFGE=PZTHEN375                                             :rem 107
475 RETURN                                                     :rem 128
480 PRINT:RETURN                                                :rem 67
485 PRINTTAB(FNR(21))"{RED}$":RETURN                           :rem 142
490 PRINTTAB(FNR(21))"{WHT}'":RETURN                           :rem 118
```

```
495 PRINTTAB(FNR(21))"{BLU}&":RETURN                    :rem 148
500 PRINTTAB(FNR(21))"{PUR}%":RETURN                      :rem 3
505 PRINTTAB(FNR(21))"{CYN}(":RETURN                     :rem 14
510 POKE646,10:PRINTTAB(FNR(21))"*":RETURN                :rem 3
515 POKE646,FNR(7)+9:PRINTTAB(FNR(21))")":RETURN        :rem 120
520 POKE646,FNR(7)+9:PRINTTAB(FNR(21))"+":RETURN        :rem 118
525 POKE646,13:FORL=1TO2:PRINTTAB(FNR(20))"{UP}"CHR$(34)CHR$(
    34):NEXT:RETURN                                      :rem 13
530 PRINTD$"{UP}";:FORL=1TO198:PRINT"{WHT} ";:NEXT:RETURN
                                                        :rem 155
535 PRINT"{CLR}";:FORP=1TO23:GOSUB480:GOSUB525:NEXT      :rem 72
540 RETURN                                              :rem 121
545 Q=0:FORP=1TO5:Q=Q+Q(P)*100*LV:NEXT                   :rem 73
550 IFQ=5000*LVTHENQ=Q+BN*100*LV:K=5                    :rem 143
555 PRINTD$"{CYN}{5 RIGHT}"RIGHT$(TI$,3),"{YEL}"Q+BO"{WHT}
                                                         :rem 93
560 FORP=1TO5:PRINTTAB(P*4-3)Q(P);:POKE7920+P*4,41-P:NEXT
                                                        :rem 221
565 POKE646,K:K=2                                       :rem 218
570 PRINTSPC(56-LEN(STR$(BN*100*LV)))"??"BN*100*LV"?? :rem 38
575 RETURN                                              :rem 129
580 DATA175,6,175,4,175,2,175,6,187,4,183,2,183,4,175,2,175,4
    ,167,2,175,8                                         :rem 80
585 DATA228,6,227,3,228,9,219,6,221,6,212,6,215,18      :rem 214
```

Steven R.
McCloskey

Hide-N-Seek

"Hide-N-Seek" is a game for two players using an unexpanded VIC and a joystick. At the start of the game, player 1 must attempt to maneuver his or her figure across the board and pick up dots to collect points. Use these keys for moving in four directions:

Up	**f5**
Down	**f7**
Right	**left/right cursor**
Left	**up/down cursor**

As the figure moves toward the opening, player 2 must try to catch and stop the figure by using the joystick to control a car. As the car moves, it puts out quad dots which are worth twice as much if player 1's character picks them up. If the character makes it to the opening on the other side of the screen, a new screen comes up with all dots increased in value. Once the car catches player 1's figure, the roles reverse: Player 2 controls the figure and uses the keyboard, while player 1 controls the car by using the joystick. When player 2 has been caught, the game is over.

Hide-N-Seek

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 POKE36879,26:POKE36878,15:CL=30720:PRINT"{CLR}{10 DOWN}
  {5 SPACES}{RED}HIDE*N*SEEK{6 SPACES}"                :rem 94
10 POKE52,28:POKE56,28:CLR:P=36874                     :rem 229
20 X=27:FORI=7168TO7679:POKEI,PEEK(I+25600):POKEP+5,X:X=X+1:I
   FX=32THENX=24                                       :rem 47
22 IFI=7200THEN600                                      :rem 7
25 NEXT:I=1:J=0:R=200                                  :rem 227
30 FORC=7432TO7487:READA:POKEC,A:NEXT:POKE36869,255    :rem 129
40 DATA195,195,170,170,170,170,195,195,235,235,40,40,40,40,23
   5,235                                               :rem 230
50 DATA0,0,0,24,24,0,0,0,0,102,102,0,0,102,102,0        :rem 11
60 DATA255,195,189,165,165,189,195,255,0,32,96,255,255,96,32,
   0,28,28,8,62,8,28,20,20                             :rem 128
65 PRINT"{CLR}{2 DOWN}{RED}PLAYER";I", THE MAN,":PRINT"{DOWN}
   USES THE KEYBOARD.":IFI=2THENA=1:GOTO70             :rem 172
67 A=2                                                  :rem 29
70 PRINT"{3 DOWN}PLAYER";A", THE CAR,":PRINT"{DOWN}USES THE J
   OYSTICK."                                           :rem 226
75 PRINT"{4 DOWN}{RVS}GET READY !!!{OFF}":FORX=1TO5000:NEXT
                                                       :rem 165
80 Y=128:CL=30720:FORX=7724TO8185:POKEX+CL,0:POKEX,35:POKEP,Y
   :Y=Y+1:IFY=149THENY=128                             :rem 234
82 NEXT:POKEP,0                                        :rem 190
85 FORX=1TO100:Y=INT(RND(1)*462)+1:POKE7723+Y+CL,2:POKE7723+Y
   ,37:POKEP+2,140+X:NEXT                              :rem 227
```

```
87 POKEP+2,0                                                    :rem 167
90 FORX=7724TO8164STEP22:POKEX+CL,2:POKEX,37:POKEX+21+CL,2:PO
   KEX+21,37:NEXT:POKE7944,32                                   :rem 150
100 FORX=7945TO7949:POKEX+CL,0:POKEX,35:POKEX+15+CL,0:POKEX+1
    5,35:NEXT:IFN=2THEN170                                      :rem 166
105 IFN=1THEN190                                                :rem 170
110 FORB=0TO96:READC:POKE831+B,C:NEXT                            :rem 45
120 DATA169,0,141,62,3                                           :rem 94
130 DATA141,60,3,141,61,3,169,0,141,19,145,169,127,141,34,145
    ,173,32,145,41,128                                           :rem 82
140 DATA201,0,208,5,169,1,141,60,3,169,255,141,34,145,173,17,
    145,41,8,201,0,208,5,169                                    :rem 116
150 DATA22,141,60,3,173,17,145,41,16,201,0,208,5,169,1,141,61
    ,3,173,17,145,41,4,201,0                                     :rem 88
160 DATA208,5,169,22,141,61,3,173,17,145,41,32,201,0,208,5,16
    9,1,141,62,3,96                                             :rem 186
170 G=0:H=10:G=G-H                                              :rem 230
190 A=7945:B=7964:C=33:D=39                                     :rem 203
200 SYS831:M=PEEK(828)-PEEK(829)                                 :rem 79
205 POKEP,0:F=0:IFPEEK(A+M)=39THEN400                          :rem 195
210 E=PEEK(197):IFE=23THENF=1                                  :rem 191
215 IFE=31THENF=-1                                               :rem 29
220 IFE=55THENF=-22                                              :rem 82
225 IFE=63THENF=22                                               :rem 41
230 IFF+B<7724ORF+B>8185THENF=0                                  :rem 49
235 IFPEEK(B+F)=37THENF=0                                      :rem 219
240 IFPEEK(B+F)=35THENG=G+H                                      :rem 96
245 IFPEEK(B+F)=36THENG=G+(H*2)                                  :rem 19
250 IFB=7944THENH=H+10:N=1:GOTO80                              :rem 214
255 POKEB,32:B=B+F:POKEB+CL,6:POKEB,39:POKEP+2,R:IFF=0THENPOK
    EP+2,0                                                     :rem 139
260 IFM=22ORM=-22THENC=34:GOTO270                              :rem 243
265 C=33                                                       :rem 131
270 IFM=21ORM=-21THENM=0                                       :rem 185
275 IFPEEK(A+M)=37THENM=0                                      :rem 236
280 IFA+M<7724ORA+M>8185THENM=0                                  :rem 73
290 POKEA+CL,0:POKEA,36:A=A+M:POKEA+CL,12:POKEA,C:POKEP+2,0:P
    OKEP,R:IFM=0THENPOKEP,0                                      :rem 19
300 PRINT"{HOME}{GRN}H.S.{BLK}";J:PRINT"{HOME}{11 RIGHT}{GRN}
    # ={BLK}";H:PRINT"{HOME}{DOWN}{GRN}PLAYER{BLK}";I"{GRN}SC
    ORE{BLK}";G                                                :rem 173
310 GOTO200                                                      :rem 95
400 GOSUB500:IFI=1THENK=G:I=2:N=2:GOTO65                       :rem 238
410 L=G:IFL>JTHENJ=L                                              :rem 4
420 IFK>JTHENJ=K                                               :rem 249
430 O=1:IFL>KTHENO=2                                           :rem 223
440 PRINT"{CLR}{3 DOWN}{RED}THE WINNER IS PLAYER";O            :rem 242
450 PRINT"{2 DOWN}HIGH SCORE";J:PRINT"{3 DOWN}{8 RIGHT}SCORE"
                                                                 :rem 79
460 PRINT"{2 DOWN}PLAYER 1 :";K:PRINT"{2 DOWN}PLAYER 2 :";L
                                                                 :rem 56
```

```
470 PRINT"{3 DOWN} HIT ANY KEY TO START"                :rem 134
480 POKE198,0:WAIT198,1:I=1:N=1:K=0:L=0:G=0:GOTO65      :rem 207
500 POKEP+4,10:FORX=230TO128STEP-1:POKEP+2,X:FORY=1TO20:NEXT:
    NEXT:POKEP+2,0:POKEP+3,200                          :rem 227
510 FORX=15TO0STEP-.05:POKEP+4,X:NEXT:POKEP+3,0:POKEP+4,15:RE
    TURN                                                :rem 147
600 PRINT"{CLR}{2 DOWN}{RED}TO MOVE WITH KEYBOARD:":PRINT"
    {DOWN}UP : PRESS F5":PRINT"{2 DOWN}DOWN : PRESS F7"
                                                        :rem 133
610 PRINT"{2 DOWN}RIGHT : PRESS LEFT -":PRINT"{DOWN}RIGHT CRS
    R"                                                  :rem 251
620 PRINT"{2 DOWN}LEFT : PRESS UP - DOWN":PRINT"CRSR":GOTO25
                                                        :rem 166
```

Steven R. McCloskey

Chopper Lift

"Chopper Lift" is a challenging game for an unexpanded VIC. It uses custom character graphics and requires a joystick. When the game begins you start with 1000 units of time. Your objective is to achieve the high score by advancing from one level to the next.

You'll start on level 1. Use a joystick to move your character back and forth across the platform, and try to catch the jumping bean. When you catch the bean, you'll receive one jump unit and 50 points.

There's a helicopter overhead with a hanging ladder. When you think you have enough jump units to reach the ladder, push the fire button. If you catch the ladder, you'll be pulled up into the helicopter and brought to the next level; you'll also receive 1000 points. If you fail to catch the ladder, you'll fall back to the platform for another try. Don't waste your jumps, for the ladder gets longer and shorter, and the platform gets smaller. If the platform disappears from underneath, you'll fall down to the next level and lose 1000 points. If you hit bottom or if time runs out, the game is over.

Chopper Lift

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 POKE36879,190:PRINT"{CLR}{11 DOWN}{5 RIGHT}{BLU}CHOPPER LIF
  T":POKE52,28:POKE56,28:CLR                          :rem 111
10 FORI=7168TO7679:POKEI,PEEK(I+25600):NEXT            :rem 98
15 FORC=7448TO7535:READA:POKEC,A:NEXT:POKE36869,255   :rem 133
20 DATA56,56,16,254,56,56,40,40,186,186,146,254,56,56,40,40,2
   4,24,16,28,16,56,104,76                            :rem 105
25 DATA48,48,16,112,16,56,44,100,0,0,0,0,0,0,24,24,24,24,0,0,
   0,0,0,0                                            :rem 235
30 DATA7,0,64,255,65,0,0,1,255,32,112,248,248,112,138,252,132
   ,252,132,252,132,252,132                           :rem 107
35 DATA252,255,255,255,255,255,255,255,255,0,0,0,0,0,16,209,2
   55                                                  :rem 73
70 PRINT"{CLR}":CL=30720:A=8021:I=21:L=26:S=16         :rem 207
75 Y=1000:V=1:W=0:Z=0:CC=36874                         :rem 201
80 POKECC+2,0:FORT=38730TO38751:POKET,4:NEXT:FORT=8033TO8052:
   POKET+CL,6:POKET,44:NEXT                            :rem 99
85 B=7789:D=22:G=B:I=I-1:L=L-1:N=8032:O=22:Q=8011:U=8010:S=S-
   1:W=0:AA=0:K=0:H=0:R=0                              :rem 29
90 FORT=38510TO38531:POKET,0:NEXT:POKE(A-22)+CL,1:POKE(A-23)+
   CL,1:GOSUB300                                      :rem 175
95 POKECC+4,15:DD=200                                 :rem 165
100 J=37151:POKEJ+3,255:POKEJ+3,127                   :rem 201
110 X=PEEK(J+1)AND128:JE=-(X=.):X=PEEK(J):JS=-((XAND8)=.):POK
    EQ,39                                             :rem 138
120 JW=-((XAND16)=.):JN=-((XAND4)=.):FB=-((XAND32)=.):rem 174
135 IFJETHENM=1:C=37:GOTO160                          :rem 207
```

```
145 IFJWTHENM=-1:C=38:GOTO160                                  :rem 16
150 IFFBTHEN600                                               :rem 114
155 M=0:C=35:                                                 :rem 177
160 POKECC,128                                                 :rem 19
165 K=K+1:IFK=LTHEN400                                         :rem 43
170 H=H+1:IFH=ITHENH=0:GOSUB300                               :rem 137
180 POKEQ,40:R=R+1:IFR=STHENGOSUB500                           :rem 15
190 IFPEEK(A+M)=40THENW=W+1:Z=Z+50:POKECC+2,DD:GOSUB500
                                                              :rem 121
230 PRINT"{HOME}{GRN}TIME{BLK}";Y"{LEFT} ":Y=Y-1:PRINT"{HOME}
    {10 RIGHT}{GRN}SCORE{BLK}";Z;"{LEFT} ":PRINT"{HOME}{DOWN}
    {GRN}JUMP{BLK}";W;"{LEFT} "                                :rem 52
240 PRINT"{HOME}{DOWN}{10 RIGHT}{GRN}LEVEL{BLK}";V;"{LEFT} "
                                                              :rem 128
245 IFY=0THEN900                                              :rem 184
250 POKEA,32:A=A+M:POKEA,C                                     :rem 34
255 POKECC,0                                                  :rem 173
260 IFPEEK(A+22)=44THEN100                                    :rem 210
265 POKEQ,32:IFV>1THEN800                                       :rem 3
270 FORT=ATOA+132STEP22:POKET+CL,4:POKET,35:POKECC+2,DD:FORR=
    1TO25:NEXT:POKET,32:NEXT                                  :rem 234
280 POKEA+154+CL,4:POKEA+154,45                               :rem 237
290 POKECC+2,0:POKECC+3,220:FORT=15TO0STEP-1:POKECC+4,T:FORX=
    1TO25:NEXTX:NEXTT:GOTO900                                     :rem 4
300 FORT=1TOF:POKEG,32:G=G-22:NEXT:POKEG,32:POKEG-1,32:rem 85
310 B=B+1:D=D-2:E=INT(RND(1)*D)+1:F=INT(RND(1)*4)+5:G=B:G=G+E
    :POKEG,42:POKEG-1,41                                      :rem 170
320 FORT=1TOF:G=G+22:POKEG,43:NEXT:RETURN                     :rem 145
400 K=0:O=O-2:POKEN+O,32:N=N+1:POKEN,32:GOTO170                :rem 68
500 R=0:POKEQ,32:P=INT(RND(1)*20)+1:Q=P+U:POKECC+2,0:RETURN
                                                               :rem 74
600 IFW=0THEN155                                              :rem 179
610 POKEA,32:POKEA+CL,1:A=A-22:POKEA,36:POKEA+CL,4:POKECC,DD:
    DD=DD+1:W=W-1:IFW=0THEN650                                 :rem 70
620 IFA<7834THEN650                                            :rem 68
630 IFPEEK(A-22)=43THENPOKEQ,32:POKECC,0:GOTO700              :rem 169
640 GOTO600                                                   :rem 105
650 POKEA,32:POKEA+CL,1:A=A+22:POKEA,36:POKEA+CL,4:POKECC,DD:
    DD=DD-1                                                   :rem 112
660 IFA>8010THEN155                                            :rem 61
670 GOTO650                                                   :rem 113
700 POKEA+CL,1:POKEA,32:A=A-22:IFPEEK(A)=42THENPOKECC+1,0:GOT
    O720                                                      :rem 174
710 POKEA+CL,4:POKEA,36:POKECC+1,DD:DD=DD+1:FORT=1TO25:NEXT:G
    OTO700                                                    :rem 179
720 POKEA,32:POKEA-1,32:A=A-22:IFA<7746THENA=A+440            :rem 140
730 POKEA+CL,0:POKE(A+CL)-1,0:POKEA,42:POKEA-1,41:POKECC+2,DD
    :DD=DD+1:FORT=1TO25:NEXT                                  :rem 173
740 IFA<8010ANDA>7987THENA=A+22:POKEA,36:V=V+1:Z=Z+1000:POKEA
    -22,32:POKEA-23,32:GOTO80                                 :rem 103
750 GOTO720                                                   :rem 110
```

```
800 FORT=1TOF:POKEG,32:G=G-22:NEXT:POKEG,32:POKEG-1,32:rem 90
810 V=V-1:Z=Z-1000                                    :rem 249
820 POKEA,32:POKEA+CL,1:A=A+22:IFA>8164THENA=A-440    :rem 178
830 IFA>8009ANDA<8032THENPOKEA+CL,4:POKEA,35:POKECC+2,0:I=I+2
    :L=L+2:S=S+2:GOTO80                                :rem 93
840 POKEA+CL,4:POKEA,35:POKECC+2,DD:DD=DD-1:FORT=1TO25:NEXT:G
    OTO820                                            :rem 188
900 POKECC+4,0:POKECC+3,0:PRINT"{HOME}{3 DOWN}{6 RIGHT}GAME
    {2 SPACES}OVER{7 SPACES}HIT ANY KEY TO PLAY!"     :rem 183
910 IFZ>HSTHENHS=Z                                    :rem 189
920 IFV>HLTHENHL=V                                    :rem 168
930 PRINT"{HOME}{RED}H.S.{BLK}";HS;"{LEFT} ":PRINT"{HOME}
    {DOWN}{RED}H.L.{BLK}";HL;"{LEFT} "                :rem 203
940 GETA$:IFA$=""THEN940                               :rem 93
950 GOTO70                                             :rem 62
```

John P. Murphy

Spring Man

"Spring Man" is a game for the unexpanded VIC. The objective is to climb up a series of ladders and platforms to reach your goal. There are six levels of difficulty. What makes the game even more complicated is that each of the platforms has five bouncing springs. The Spring Man must climb ladders and run across the platforms to reach the top. You score points through bonus points that are earned as each level is completed.

On each level, bonus points tick away as time passes. Once all the points are gone, the Spring Man loses all his lives and the game ends. The Spring Man can lose a life in various ways—by falling off the ladders or platforms, by running out of time, or by being smashed by the springs.

The joystick can make three different movements: left, right, and up (climbing the ladders).

This is not an easy game to win. As a reward for your skillfull play, you win another life each time you achieve a new level (four lives are given per game). In addition, the game speeds up after the third level.

Altering the Program

To make the game easier, you may want to increase the value of N in line 14. Also in line 14, you can change the variable CO to equal a higher number by adding 16 for every level you want to advance. This number should never exceed 136.

Spring Man

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
5 POKE52,28:POKE56,28:POKE51,0:POKE55,0:CLR:FORX=7424TO7432:P
  OKEX,0:NEXT                                          :rem 208
7 FORX=7168TO7255:READDA:POKEX,DA:NEXT:POKE36869,255    :rem 6
14 PRINT"{CLR}":N=4:CO=56                              :rem 54
15 PRINT"{HOME}{CLR}{RED}{3 DOWN}{2 SPACES}@@@@@@@@@@@@@@@@@@@
   @@":PRINT"{DOWN}{2 SPACES}@@@@@@@@@@@@@@@@@@"      :rem 200
20 PRINT"{2 DOWN}{3 SPACES}@@@@@@@@@@@@@@@@@@":PRINT"{2 DOWN}
   {3 SPACES}@@@@@@@@@@@@@@@"                          :rem 130
25 PRINT"{2 DOWN} @@@@@@@@@@@@@@@@@@@":PRINT"{2 DOWN} @@@@@@@@
   @@@@@@@@@@@ "                                       :rem 135
30 PRINT"{2 DOWN}@@@@@@@@@@@@@@@@@@@@                   :rem 52
40 PRINT"{HOME}";SPC(17)"{OFF}{WHT}E{RVS} -";N;"{HOME}{DOWN}"
   SC                                                  :rem 67
45 IFN<=0THEN300                                       :rem 177
82 POKE36879,CO:LA=7745                                 :rem 67
84 IFCO>136THEN910                                      :rem 46
85 J2=37137:J3=37152:J4=28:J5=128:J6=37154:POKEJ6,127:rem 187
86 T=32:W=6:G=3:H=2:K=1:AA=23:BB=21:LL=22:P=197        :rem 212
```

```
87 M=8120:POKE36878,15:POKEM,10:S1=36877:BO=9000:S2=36874:S3=
   36876:CL=104                                          :rem 155
88 F1=4:F2=5:F3=7:F4=8:F6=10:O=0:BI=100:L1=20:L2=16:L3=19:S7=
   200:CI=136:S8=212                                     :rem 130
89 RI=INT(RND(1)*2):A=7900:C=8055:B=8009:D=7878:E=7790:L=230
                                                         :rem 105
90 A=A+RI:C=C+RI:B=B-RI:D=D-RI:E=E-RI                      :rem 4
91 FORU=1TO11                                             :rem 25
92 POKEA,G:POKEB,W:POKEC,W:POKED,W:POKEE,W                :rem 12
93 IFPEEK(M)=3ORPEEK(M)=6THEN930                         :rem 208
94 J=(PEEK(J3)ANDJ5)OR(PEEK(J2)ANDJ4):J=ABS((J-BI)/4)-7
                                                         :rem 193
95 IFJ=11THENPOKEM,T:M=M+K:POKEM,F1:POKES2,S7:POKES2,O
                                                         :rem 254
96 IFJ=3THENPOKEM,T:M=M-K:POKEM,F3:POKES2,S7:POKES2,O:rem 212
98 IFJ=6ANDPEEK(M)=KORPEEK(M)=HTHENPOKEM,T:M=M-LL:POKEM,F6:PO
   KES2,S7:POKES2,O                                       :rem 48
99 POKEA,T:POKEB,T:POKEC,T:POKED,T:POKEE,T                :rem 20
100 A=A+AA:B=B-AA:C=C-BB:D=D-AA:E=E-BB:POKES3,S8          :rem 91
102 IFM=LATHEN940                                          :rem 5
103 IFCO=CIANDPEEK(LA)=TTHENPOKELA,9                      :rem 53
104 PRINT"{HOME}{BLK}{3 DOWN}{2 SPACES}B{DOWN}{LEFT}A{DOWN}
    {LEFT}A";TAB(L1)"{DOWN}B{DOWN}{LEFT}A{DOWN}{LEFT}A"
                                                         :rem 177
105 PRINT"{3 SPACES}B{DOWN}{LEFT}A{DOWN}{LEFT}A";TAB(L2)"
    {DOWN}B{DOWN}{LEFT}A{DOWN}{LEFT}A"                   :rem 221
106 PRINT" B{DOWN}{LEFT}A{DOWN}{LEFT}A";TAB(L3)"{DOWN}B{DOWN}
    {LEFT}A{DOWN}{LEFT}A"                                :rem 223
109 POKEA,W:POKEB,G:POKEC,G:POKED,G:POKEE,G                :rem 11
110 IFPEEK(M+LL)=TTHENN=N-1:GOTO500                      :rem 166
111 IFPEEK(M)=3ORPEEK(M)=6THEN930                        :rem 247
112 J=(PEEK(J3)ANDJ5)OR(PEEK(J2)ANDJ4):J=ABS((J-BI)/4)-7
                                                         :rem 232
113 IFJ=11THENPOKEM,T:M=M+K:POKEM,F2:POKES2,S7:POKES2,O
                                                          :rem 38
114 IFJ=3THENPOKEM,T:M=M-K:POKEM,F4:POKES2,S7:POKES2,O
                                                         :rem 252
115 IFJ=6ANDPEEK(M)=KORPEEK(M)=HTHENPOKEM,T:M=M-LL:POKEM,F6:P
    OKES2,S7:POKES2,O                                     :rem 86
119 IFCO<CLTHENFORI=1TOBI:NEXT                           :rem 251
120 POKEA,T:POKEB,T:POKEC,T:POKED,T:POKEE,T               :rem 53
125 BO=BO-BI                                             :rem 175
130 A=A-BB:B=B+BB:C=C+AA:D=D+BB:E=E+AA:POKES3,O           :rem 30
135 IFM=LATHEN940                                         :rem 11
140 PRINT"{HOME}{WHT}{RVS}"BO:IFBO=<OTHENN=0:GOTO300      :rem 51
150 NEXTU                                                 :rem 42
160 GOTO89                                                :rem 65
200 DATA255,24,36,66,255,0,0,0,36,60,36,60,36,60,36,60,231,60
    ,36,60,231,60,36,60                                  :rem 117
210 DATA254,56,84,84,84,84,56,254,3,3,12,10,152,100,4,8,6,6,5
    2,73,22,48,72,132                                     :rem 40
```

```
220 DATA0,254,56,84,146,84,56,255,96,96,44,146,104,12,18,33,1
    92,192,48,80,25,38,32,16                          :rem 164
230 DATA16,56,16,56,84,56,124,40,16,56,16,56,84,16,40,40
                                                      :rem 248
300 POKES1,220:FORL=15TO0STEP-1:POKES4,L:FORM=1TO100:NEXTM:NE
    XTL:POKES1,0:POKES4,0                              :rem 27
305 IFN>=1ANDCO<=136THEN15                             :rem 46
310 PRINT"{HOME}{10 DOWN}"SPC(5)"{RVS}{BLK}PLAY AGAIN?":POKES
    3,0                                               :rem 148
312 GETA$:IFA$="Y"THEN5                                :rem 71
313 IFA$="N"THENSYS65234                              :rem 140
314 GOTO312                                           :rem 103
500 POKES3,0:L=L-1:POKES3,L:POKEM,T:M=M+LL:POKEM,4    :rem 154
505 IFPEEK(M+LL)=0ORM>8185THENPOKES3,0:GOTO300        :rem 100
510 GOTO500                                           :rem 100
910 EP=135:FORI=1TO106:POKES3,0:EP=EP+1:POKES3,EP:NEXT:POKES3
    ,0                                                :rem 211
920 PRINT"{HOME}{10 DOWN}{RVS}{6 SPACES}GOOD GAME":FORJ=1TO10
    000:NEXT:SYS65234                                 :rem 161
930 POKES1,128:FORI=1TO100:NEXT:POKES1,0:N=N-1:GOTO15 :rem 46
940 SC=SC+BO:CO=CO+16:POKES3,0:N=N+1:GOTO15           :rem 130
```

Larry
Volz

Trench Wars

"Trench Wars" is a fast-moving, reverse-scrolling action game of skill for the unexpanded VIC and a joystick.

You are the pilot of a cargo ship that, in accordance with the great convention against weaponry, is unarmed. Your goal is to maneuver within the trench and pick up cargo while avoiding the kamikaze Aliens (they too are unarmed). You start with three men and a choice of nine levels. After every minute of play, the game goes to a docking subroutine which will add 50 points to your score and advance you to the next level of difficulty. Do *not* hit the trench walls. Each cannister of cargo is worth 1 point, and you get an extra man every 150 points.

The Program

The play is rather simple, but it is trickier than it seems. The program reverse scrolls everything on the screen except the title and the player's ship. It makes extensive use of the programmable characters. The number of characters on the screen at any time is determined in line 58. The probability of a character being printed each line is determined by multiplying the level number by 10 percent. Eventually, there is a character on every line, and it seems that "joystick wrist" becomes more dangerous than the Aliens.

I have included a rather unusual explosion feature that makes use of the programmable characters. This lies within the subroutine starting at 8000. It first POKEs 30 (the player ship) into the screen location G and blanks out the space above it (G−22: whatever was run into). It then randomly changes the statistics for this character and gives an illusion of flame. Next, 2 is POKEd into the same location, which is a previously compiled character to look like debris. All this done in conjunction with the white noise generator explosion gives an excellent effect. Unfortunately, the statistics for the player's ship must then be recompiled (this explains the seemingly useless loop at 9999).

The subroutine at 5000 checks to see if the joystick has been moved right or left only. JS(X+1,Y+1) will return either a 2 if the joystick is right or a 6 if the joystick is left. Once the operation is performed in line 5000, R will equal a number that is either positive or negative, and the movement is based on that number.

If the ship goes far enough in either direction to hit the wall, it goes to the subroutine that starts at 8000 (this is determined in lines 5000 and 5005).

Trench Wars

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
Ø GOSUB9999:POKE36879,8:AP=15Ø                    :rem 16
1 PRINT"{CLR}{4 SPACES}TRENCH WARS"              :rem 15Ø
2 PRINT"DIFFICULTY (1-9)?"                        :rem 29
```

```
3 GETA$:A=VAL(A$):IFA=0ORA<0ORA>9THEN3                          :rem 73
4 PRINT"{UP}{20 SPACES}":PRINT"{UP}{RED}!!!!!!!{10 SPACES}!!!!
  !!"                                                          :rem 218
5 G=8098+9:C=38818+9                                            :rem 32
7 PM=3                                                          :rem 68
8 POKE36878,5:POKE36877,255                                     :rem 24
10 DIM JS(2,2):POKE37139,0:DD=37154:PA=37137:PB=37152:rem 160
20 FORI=0TO2:FORJ=0TO2:READJS(J,I):NEXTJ,I                      :rem 188
40 TI$="000000"                                                 :rem 198
49 PRINT"{HOME}{CYN}";PM;" TRENCH WARS";PP                      :rem 35
50 GOSUB9000:PRINT"{HOME}{DOWN}{RED}!!!!!!!{9 SPACES}!!!!!!!"
                                                                :rem 166
51 PRINT"{HOME}{DOWN}{LEFT}{INST}":POKE218,158                  :rem 90
55 POKEG,30:POKEC,INT(RND(1)*6+1)                               :rem 165
56 R=INT(RND(1)*2):R1=INT(RND(1)*200+1):IFR=0THENZ$="{WHT}"
                                                                :rem 103
57 IFR=1THENZ$="{BLU}"                                          :rem 8
58 IFR1>100-(A*10)THENPRINT"{HOME}{2 DOWN}";TAB(INT(RND(1)*9+
   6));Z$;CHR$(R+64)                                            :rem 195
59 POKEG,32                                                     :rem 117
60 IFJS(X+1,Y+1)<>8THENGOSUB5000                                :rem 158
65 IFPEEK(G-22)<>32THEN8000                                     :rem 30
70 IFVAL(TI$)>100THENGOSUB500                                   :rem 254
80 GOTO50                                                       :rem 6
100 POKE36878,15:POKE36874,240:POKE36878,5:POKE36874,0:rem 68
110 IFPP>APTHENGOSUB200:GOTO49                                  :rem 177
120 RETURN                                                      :rem 115
200 AP=AP+150:PM=PM+1:FORT=1TO2:POKE36877,0:POKE36878,15
                                                                :rem 86
210 POKE36876,220:FORH=1TO200:NEXT:POKE36876,240:FORH=1TO200:
    NEXT:POKE36876,0                                            :rem 166
220 NEXTT:POKE36878,5:POKE36877,255:RETURN                      :rem 91
500 DO$=";<{DOWN}{2 LEFT}=>"                                    :rem 10
501 POKEG,32:G=8098+9:C=38818+9                                 :rem 194
505 FORT=1TO20                                                  :rem 72
506 PRINT"{HOME}{DOWN}{LEFT}{INST}":POKE218,158                 :rem 143
507 POKEG,30:POKEC,6                                            :rem 175
510 PRINT"{DOWN}{RED}!!!!!!!{9 SPACES}!!!!!!!"                  :rem 65
511 FORH=1TO50:NEXT                                             :rem 181
515 POKEG,32                                                    :rem 162
520 NEXTT                                                       :rem 42
540 PRINTTAB(9);DO$                                             :rem 121
550 FORT=1TO12                                                  :rem 73
556 PRINT"{HOME}{DOWN}{LEFT}{INST}":POKE218,158                 :rem 148
557 POKEG,30:POKEC,6                                            :rem 180
560 PRINT"{DOWN}{RED}!!!!!!!{9 SPACES}!!!!!!!"                  :rem 70
571 FORH=1TO50:NEXT                                             :rem 187
575 POKEG,32                                                    :rem 168
580 NEXTT                                                       :rem 48
590 POKEG,30:POKE36878,15:POKE36877,0:FORT=1TO2                 :rem 58
```

```
600 POKE36876,240:FORH=1TO150:NEXT:POKE36876,0:FORH=1TO10
                                                    :rem 204
610 NEXTH,T:PP=PP+50:POKEG,32:POKE36878,5:POKE36877,255
                                                     :rem 64
620 FORT=1TO7:PRINT"{HOME}{DOWN}{LEFT}{INST}":POKE218,158
                                                     :rem 73
625 PRINT"{DOWN}{RED}!!!!!!!{9 SPACES}!!!!!!!"       :rem 72
626 FORH=1TO50:NEXT                                 :rem 188
630 NEXTT                                            :rem 44
635 IFPP>APTHENGOSUB200                             :rem 221
1000 PRINT"{HOME}{4 SPACES}TRENCH WARS{7 SPACES}":TI$="000000
     ":A=A+1                                        :rem 151
1001 RETURN                                         :rem 162
5000 R=JS(X+1,Y+1)-4:IFR<0ANDG+1>=8098+15THEN8000    :rem 17
5005 IFR>0ANDG-1<=8098+5THEN8000                     :rem 58
5010 IFR<0THENG=G+1:C=C+1:RETURN                    :rem 220
5020 IFR>0THENG=G-1:C=C-1:RETURN                    :rem 227
5030 RETURN                                         :rem 168
8000 REM BOOM                                       :rem 217
8010 IFPEEK(G-22)=0THENPP=PP+1:GOSUB100:GOTO49       :rem 70
8020 POKEG,30:FORT=15TO0STEP-3:POKE36878,T:POKE36877,255
                                                    :rem 143
8021 POKEG-22,32:FORX=7168+(8*30)TO7168+(8*30)+7:POKEX,INT(RN
     D(1)*256):NEXTX                                :rem 115
8022 NEXTT:POKE36878,5:POKEG,2                       :rem 71
8035 RESTORE:FORZ=7168+(8*30)TO7168+(8*30)+7:READZZ:POKEZ,ZZ:
     NEXTZ                                           :rem 95
8036 POKEG,32:G=8098+9:C=38818 +9                   :rem 253
8990 PM=PM-1:IFPM=0THENPRINT"{CLR}{7 DOWN}{3 SPACES}GAME OVER
     ";PP:POKE36877,0                               :rem 145
8991 IFPM=0THENFORT=1TO1000:NEXT:RUN                 :rem 88
8999 POKE646,6:GOTO49                               :rem 244
9000 POKEDD,127:S3=-((PEEK(PB)AND128)=0):POKEDD,255  :rem 174
9010 P=PEEK(PA):S1=-((PAND8)=0):S2=((PAND16)=0):S0=((PAND4)=0
     )                                              :rem 229
9020 FR=-((PAND32)=0):X=S2+S3:Y=S0+S1:RETURN        :rem 133
9999 PRINT"{CLR}":FORT=1TO8:READZ$:NEXT              :rem 83
10000 POKE52,28:POKE56,28:FORI=7168TO7679:POKEI,PEEK(I+25600)
      :NEXT:POKE36869,255                             :rem 3
10020 FORC=7168+(8*30)TO7168+(8*30)+7:READA:POKEC,A:NEXT
                                                    :rem 182
10030 FORC=7432TO7432+7:READA:POKEC,A:NEXT           :rem 41
10040 FORC=7168+16TO7168+16+7:READA:POKEC,A:NEXT     :rem 90
10050 FORC=7168+8TO7168+15:READA:POKEC,A:NEXT       :rem 201
10060 FORC=7168TO7168+7:READA:POKEC,A:NEXT           :rem 56
10070 FORC=7168+(8*59)TO7168+(8*59)+7:READA:POKEC,A:NEXT
                                                    :rem 209
10080 FORC=7168+(8*60)TO7168+(8*60)+7:READA:POKEC,A:NEXT
                                                    :rem 194
10090 FORC=7168+(8*61)TO7168+(8*61)+7:READA:POKEC,A:NEXT
                                                    :rem 197
```

```
10100 FORC=7168+(8*62)TO7168+(8*62)+7:READA:POKEC,A:NEXT
                                                    :rem 191
19998 PRINT"{CLR}":RETURN                           :rem 146
19999 DATA16,16,16,84,186,146,186,186               :rem 124
20000 DATA16,16,16,84,186,146,186,186                :rem 89
20010 DATA255,255,255,255,255,255,255,255             :rem 33
20020 DATA129,42,68,145,4,100,10,160                  :rem 9
20030 DATA40,108,108,16,56,84,16,16                 :rem 229
20040 DATA126,126,126,60,126,126,126,0              :rem 112
20050 DATA16,56,16,16,25,23,18,23                   :rem 127
20060 DATA8,28,8,8,152,232,200,232                  :rem 176
20070 DATA23,18,23,25,16,16,56,16                   :rem 129
20080 DATA232,200,232,152,8,8,28,8                  :rem 178
29999 DATA7,0,1,6,8,2,5,4,4                         :rem 101
```

Appendices

Appendix A

Beginner's Guide to Typing In Programs

What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential,* but without a program, it isn't going anywhere. Most of the programs published in this book are written in a computer language called BASIC. BASIC is easy to learn and is built into all VIC-20s.

BASIC Programs

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one right way of stating something. Every letter, character, and number is significant. A common mistake is substituting a letter such as *O* for the numeral *0,* a lowercase *l* for the numeral *1,* or an uppercase *B* for the numeral *8.* Also, you must enter all punctuation marks, such as colons and commas, just as they appear in the book. Spacing can be important. To be safe, type in the listings *exactly* as they appear.

Braces and Special Characters

The exception to this typing rule is when you see the braces, such as {DOWN}. Anything within a set of braces is a special character, or characters, that cannot easily be listed on a printer. When you come across such a special statement, refer to "How to Type In Programs" (Appendix B).

About DATA Statements

Some programs contain a section, or sections, of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (in machine language), while others may contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up, or crash. The keyboard and RUN/STOP key may seem dead, and the screen may go blank. But don't panic. No damage has been done. To regain control, turn off your computer and then turn it back on. This will erase whatever program was in memory, *so always save a copy of your program before you run it.* If your computer crashes, you can load the program and look for your mistake.

Sometimes a mistyped DATA statement will cause an error message when the program is run. The error message may refer to the program line that READs the data. *However, the error is still in the DATA statements.*

Appendix A

Get to Know Your Machine

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program so that you won't have to type it in every time you want to use it. Learn to use your machine's editing functions. How do you change a line if you made a mistake? You can always retype the line, but you should at least know how to backspace. Do you know how to enter reverse-video, lowercase, and control characters? It's all explained in your manual.

In order to insure accurate entry of each program line, we have included a checksum program. Please read "The Automatic Proofreader" (Appendix C) before typing in any of the programs in this book.

A Quick Review

1. Type in the program a line at a time in order. Press RETURN at the end of each line. Use the INST/DEL key to correct mistakes.
2. Check the line you've typed against the line in the book. You can check the entire program again if you get an error when you run the program.

Appendix B

How to Type In Programs

Many of the programs in this book contain special control characters (for example, cursor controls, color keys, and reverse video). To make it easy to know exactly what to type when entering one of these programs into your computer, we have established the following listing conventions.

Generally, VIC-20 program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor-down key. {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the SHIFT key. This would appear on your screen as a heart symbol. If you find an underlined key enclosed in braces (for example, {10 N}), you should type the key as many times as indicated. In that case, you would enter ten shifted N's.

If a key is enclosed in special brackets, [<>], you should hold down the *Commodore key* while pressing the key inside the special brackets. (The Commodore key is the key in the lower-left corner of the keyboard.) Again, if the key is preceded by a number, you should press the key as many times as necessary.

Keyword Abbreviations

In order to get as much as possible into an unexpanded VIC-20, many programmers pack program lines by leaving out spaces wherever possible and by using keyword abbreviations when writing the program. One VIC program line (a logical line) can have a maximum of 88 characters including spaces—four screen (physical) lines. By using abbreviations a programmer can save memory by putting more code on a logical line than would be possible if each keyword were spelled out.

The listings in this book contain all the keywords spelled out. If you try to enter a program line with more than 88 characters, the line will not be saved in the VIC's memory completely, and the program will not function correctly. In order to enter lines that otherwise would be too long, it's necessary to use keyword abbreviations. Below is a list of some of the more frequently used keywords with their abbreviations; for a complete list see Appendix D of *Personal Computing on the VIC-20*, the manual that came with the computer.

Keyword	Abbreviation
CHR$	C SHIFT-H
DATA	D SHIFT-A
FOR	F SHIFT-O
NEXT	N SHIFT-E
PEEK	P SHIFT-E
POKE	P SHIFT-O
PRINT	?
RETURN	RE SHIFT-T
TAB(	T SHIFT-A
THEN	T SHIFT-H

Quote Mode

You know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {DOWN}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT-2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the INST/DEL key; you can still use INST/DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is just to press RETURN. You'll then be out of quote mode and can cursor up to the mistyped line and fix it.

In order to insure accurate entry of each program line, we have included a checksum program. Please read "The Automatic Proofreader" (Appendix C) before typing in any of the programs in this book.

Refer to the following table when entering cursor and color control keys:

When You Read:	Press:		See:
{CLR}	SHIFT	CLR/HOME	
{HOME}		CLR/HOME	
{UP}	SHIFT	↑ CRSR ↓	
{DOWN}		↑ CRSR ↓	
{LEFT}	SHIFT	← CRSR →	
{RIGHT}		← CRSR →	
{RVS}	CTRL	9	
{OFF}	CTRL	0	
{BLK}	CTRL	1	
{WHT}	CTRL	2	
{RED}	CTRL	3	
{CYN}	CTRL	4	
{PUR}	CTRL	5	

When You Read:	Press:		See:
{GRN}	CTRL	6	
{BLU}	CTRL	7	
{YEL}	CTRL	8	
{ F1 }		f1	
{ F2 }	SHIFT	f1	
{ F3 }		f3	
{ F4 }	SHIFT	f3	
{ F5 }		f5	
{ F6 }	SHIFT	f5	
{ F7 }		f7	
{ F8 }	SHIFT	f7	
←	←		
↑	SHIFT	↑	

Appendix C

The Automatic Proofreader

Charles Brannon

"The Automatic Proofreader" will help you type in program listings without typing mistakes. It is a short error-checking program that hides itself in memory. When activated, it lets you know immediately after typing a line from a program listing if you have made a mistake. Please read these instructions carefully before typing any programs in this book.

Preparing the Proofreader

1. Using the listing below, type in the Proofreader. Be very careful when entering the DATA statements—don't type an *l* instead of a *1*, an *O* instead of a *0*, extra commas, and so forth.
2. Save the Proofreader on tape or disk at least twice *before running it for the first time.* This is very important because the Proofreader erases part of itself when you first type RUN.
3. After the Proofreader is saved, type RUN. It will check itself for typing errors in the DATA statements and warn you if there's a mistake. Correct any errors and save the corrected version. Keep a copy in a safe place—you'll need it again and again, every time you enter a program from this book, *COMPUTE!'s Gazette,* or *COMPUTE!* magazine.
4. When a correct version of the Proofreader is run, it activates itself. You are now ready to enter a program listing. If you press RUN/STOP–RESTORE, the Proofreader is disabled. To reactivate it, just type the command SYS 886 and press RETURN.

Using the Proofreader

Most of the listings in this book have a *checksum number* appended to the end of each line, for example, *:rem 123. Don't enter this statement when typing in a program.* It is just for your information. The rem makes the number harmless if someone does type it in. It will, however, use up memory if you enter it, and it will confuse the Proofreader, even if you entered the rest of the line correctly.

When you type in a line from a program listing and press RETURN, the Proofreader displays a number at the top of your screen. *This checksum number must match the checksum number in the printed listing.* If it doesn't, it means you typed the line differently from the way it is listed. Immediately recheck your typing. Remember, don't type the rem statement with the

checksum number; it is published only so that you can check it against the number which appears on your screen.

The Proofreader is not picky about spaces. It will not notice extra spaces or missing ones. This is for your convenience since spacing is generally not important. But occasionally proper spacing *is* important, so be extra careful with spaces.

Due to the nature of a checksum, the Proofreader will not catch all errors. Since 1 + 3 + 5 = 3 + 1 + 5, the Proofreader cannot catch errors of transposition. Thus, the Proofreader will not notice if you type GOTO 385 where you mean GOTO 835. In fact, you could type in the line in any order and the Proofreader wouldn't notice. The Proofreader should help you catch most typing mistakes, but keep this in mind if a program that checks out with the Proofreader still seems to have errors.

There's another thing to watch out for: If you enter the line by using abbreviations for commands, the checksum will not match up. But there is a way to make the Proofreader check it. After entering the line, LIST it. This eliminates the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way. *Do not use this method if the line required the use of abbreviations in order to fit within the 88-character limit.*

Special Tape SAVE Instructions

When you're through typing in a listing, you must disable the Proofreader before saving the program on tape. Disable the Proofreader by pressing RUN/STOP–RESTORE (hold down the RUN/STOP key and sharply hit the RESTORE key). This procedure is not necessary for disk SAVEs, *but you must disable the Proofreader this way before a tape SAVE.*

SAVE to tape erases the Proofreader from memory, so you'll have to load and run it again if you want to type another listing. SAVE to disk does not erase the Proofreader.

Hidden Perils

The Proofreader's home in memory is not a very safe haven. Since the cassette buffer is wiped out during tape operations, you need to disable the Proofreader with RUN/STOP–RESTORE before you save your program. This applies only to tape use. Disk users have nothing to worry about.

Not so for VIC-20 owners with tape drives. What if you type in a program in several sittings? The next day, you come to your computer, load and run the Proofreader, then try to load the partially completed program so that you can add to it. But since the Proofreader is trying to hide in the cassette buffer, it is wiped out!

What you need is a way to load the Proofreader after you've loaded the partial program. The problem is, a tape LOAD to the buffer destroys what it's supposed to load.

If you intend to type in a program in more than one sitting or wish to make a safety SAVE, follow this procedure:

1. Load and run the Proofreader.
2. Disable it by pressing RUN/STOP–RESTORE.
3. Type the following three lines in direct mode (without line numbers):

```
A$="PROOFREADER.T":B$="{10 SPACES}":FOR X=1 TO 4:A
  $=A$+B$:NEXT X

FOR X=886 TO 1018:A$=A$+CHR$(PEEK(X)):NEXT X

OPEN 1,1,1,A$:CLOSE 1
```

After you enter the last line, you will be asked to press RECORD and PLAY on your cassette recorder. Put this program at the beginning of a new tape.

You now have a new version of the Proofreader. Turn your computer off and on, then load the program you were working on. Put the cassette containing the Proofreader into the tape unit and type:

OPEN1:CLOSE1

You can now start the Proofreader by typing SYS 886. To test this, PRINT PEEK (886) should return the number 173. If it does not, repeat the steps above, making sure that A$ ("PROOFREADER.T") contains 13 characters and that B$ contains 10 spaces.

You can now reload the Proofreader into memory whenever LOAD or SAVE destroys it, restoring your personal typing helper.

The Automatic Proofreader

```
100 PRINT"{CLR}PLEASE WAIT...":FORI=886TO1018:READA:CK=CK+A:P
    OKEI,A:NEXT
110 IF CK<>17539 THEN PRINT"{DOWN}YOU MADE AN ERROR":PRINT"IN
     DATA STATEMENTS.":END
120 SYS886:PRINT"{CLR}{2 DOWN}PROOFREADER ACTIVATED.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
```

```
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

To order your copy of the VIC-20 Collection Disk call our toll-free US order line: 1-800-334-0868 (in NC call 919-275-9809) or send your prepaid order to:

VIC-20 Collection Disk
COMPUTE! Publications
P.O. Box 5058
Greensboro, NC 27403

All orders must be prepaid (check, charge, or money order). NC residents add 4.5% sales tax.

Send _____ copies of the VIC-20 Collection Disk at $12.95 per copy.

Subtotal $_____

Shipping & Handling: $2.00/disk $_____

Sales tax (if applicable) $_____

Total payment enclosed $_____

☐ Payment enclosed
Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No. _____ Exp. Date _____
Required

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4-5 weeks for delivery.

4580073

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!'s Gazette** for Commodore.

For Fastest Service
Call Our **Toll-Free** US Order Line
800-334-0868
In NC call 919-275-9809

COMPUTE!'s Gazette
P.O. Box 5058
Greensboro, NC 27403

My computer is:
☐ Commodore 64 ☐ VIC-20 ☐ Other_____

☐ $24 One Year US Subscription
☐ $45 Two Year US Subscription
☐ $65 Three Year US Subscription

Subscription rates outside the US:

☐ $30 Canada
☐ $65 Air Mail Delivery
☐ $30 International Surface Mail

Name _____
Address _____
City _____ State _____ Zip _____
Country _____

Payment must be in US funds drawn on a US bank, international money order, or charge card. Your subscription will begin with the next available issue. Please allow 4–6 weeks for delivery of first issue. Subscription prices subject to change at any time.

☐ Payment Enclosed ☐ Visa
☐ MasterCard ☐ American Express

Acct. No. _____ Expires ____/____
(Required)

The *COMPUTE!'s Gazette* subscriber list is made available to carefully screened organizations with a product or service which may be of interest to our readers. If you prefer not to receive such mailings, please check this box ☐.

758199

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **800-334-0868** (in NC 919-275-9809) or write COMPUTE! Books, P.O. Box 5058, Greensboro, NC 27403.

Quantity	Title	Price*	Total
_____	SpeedScript: The Word Processor for the Commodore 64 and VIC-20 (94-9)	**$ 9.95**	_____
_____	Commodore SpeedScript Book Disk	**$12.95**	_____
_____	COMPUTE!'s Commodore 64/128 Collection (97-3)	**$12.95**	_____
_____	All About the Commodore 64, Volume Two (45-0)	**$16.95**	_____
_____	All About the Commodore 64, Volume One (40-X)	**$12.95**	_____
_____	Programming the Commodore 64: The Definitive Guide (50-7)	**$19.95**	_____
_____	COMPUTE!'s Data File Handler for the Commodore 64 (86-8)	**$12.95**	_____
_____	Kids and the Commodore 64 (77-9)	**$12.95**	_____
_____	COMPUTE!'s Commodore Collection, Volume 1 (55-8)	**$12.95**	_____
_____	COMPUTE!'s Commodore Collection, Volume 2 (70-1)	**$12.95**	_____
_____	COMPUTE!'s VIC-20 and Commodore 64 Tool Kit: BASIC (32-9)	**$16.95**	_____
_____	Programming the VIC (52-3)	**$24.95**	_____
_____	VIC Games for Kids (35-3)	**$12.95**	_____
_____	COMPUTE!'s First Book of VIC (07-8)	**$12.95**	_____
_____	COMPUTE!'s Second Book of VIC (16-7)	**$12.95**	_____
_____	COMPUTE!'s Third Book of VIC (43-4)	**$12.95**	_____
_____	Mapping the VIC (24-8)	**$14.95**	_____
_____	COMPUTE!'s VIC-20 Collection (007)	**$12.95**	_____

*Add $2.00 per book for shipping and handling.
Outside US add $5.00 air mail or $2.00 surface mail.
NC residents add 4.5% sales tax.

Shipping & handling: $2.00/book _____
Total payment _____

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.

☐ Payment enclosed.

Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No._____ Exp. Date_____

Name_____

Address_____

City_____ State _____ Zip_____

*Allow 4–5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

4580073

Index

AI. *See* artificial intelligence
aligning numbers 20
"Alternating" program 40
analog/digital converter 87
AND# ML instruction 70
ANDZ ML instruction 70
arcade games 38
arcade-style subroutines 38–40
array 65
artificial intelligence 220–24
ASLA ML instruction 70
assembler 48
bar graph 130–31
"Base Defense" program 239–40
BASIC
  keywords 54–55
  start of, changing 5
"Battleship" program 199–202
"Big Screen Print" program 62–63
bit 3
bitmapping 91, 128, 254
"Bitmapping Pixel" program 128–29
block of memory, saving to tape 13–14
"Bomber Squadron" program 241–45
"Boogieball" program 296–99
border color 135
chaining programs 24–25
"Challenger One" program 273–75
character sets, memory locations for control of 6
"Characters" program 79
"Chopper Lift" program 315–17
CLC ML instruction 70
CLOSE command 12
CMP ML instruction 70
CMP# ML instruction 70
"Code Game" program 189–90
Commodore printer, "Write-On" and 155
compressed BASIC keywords 54–56
"Compressed List" program 56
COMPUTE!'s First Book of VIC 50
COMPUTE!'s Mapping the VIC 72
COMPUTE!'s Second Book of VIC 41
COMPUTE!'s VIC-20 and Commodore 64 Took Kit 72
custom characters 5–8, 38, 78–80, 91, 178–79, 241
DATA statement 158, 327
DECZ ML instruction 70
"Diamond Maze" program 215–16
disassembler 48–49, 51–52
"Disassembler" program 51–52
disk 10, 44–45
dummy records, relative files and 12
"Dungeon Escape" program 286–90
"8K Setup" program (VIC graph) 93
"Example 1" program (multiprocessing) 61
"Example 2" program (multiprocessing) 61
"Ex-blast" program 246–48
expansion cartridge 3, 5, 46, 120, 135, 138, 167–68, 254, 264, 279, 300
"Experiment, The" program 9
"Fasplot" program 82–83
1541 model disk drive 10
fire button 34
"Flag" program 136
"Flipper" program 203–4
FOR-NEXT loop 23, 130
game paddles 87–90, 251
GET statement 21
"Graph Expanded" program (VIC Graph) 93
graphic pictures 105–7
graphics characters, mixing with text 9
"Graph" program 130–34
"Graph Unexpanded" program (VIC Graph) 93–94
"Heist" program 211–14
"Hide-N-Seek" program 312–14
high VIC 5, 6
"Hyper Ballshot" program 251–53
input 18–22
IRQ interrupt 57, 58, 159
JMP ML instruction 70
"Joyfast" program 82
joystick 34, 36, 81–82, 178, 185, 195, 211, 215, 217, 235, 249, 258, 267, 300, 303, 312, 321
KernaL 41, 48
"Keytest" program 50, 52–53
keywords, BASIC 54–55
"Lap Racer" program 178–81
"Laser Command" program 254–57
"Laser War" program 270–72
LDA ML instruction 49, 70
LDX# ML instruction 70
LDXY ML instruction 70
"Light Cycles" program 300–302
LOAD command 13–16
logical file number 10, 11
logical line 54
"Logicolor" program 191–94
low VIC 5, 6
LSRA ML instruction 70
LSR ML instruction 70
machine language 70–73
Machine Language for Beginners 71
"Maneuver" program 195–98
"Marquee Demo" program 159–61
masking 21–22
"Mate" program (Music Mate) 117–19
mazes 220–24

maze strategies 221–23
"Meltdown" program 217–19
memory address, loading to 1–15
memory expansion 5
memory pointers 5
"Message Board" program 162–63
"Meteors" program 249–50
"Microassembler" 70–77
"Move X" program 34–35
"Moving" program 40
multiprocessing 57–61
"Multiprocess" program 60–61
music 115–27
"Music Mate" program 116
"Music Maker" program 140–48
"Music Mate" system 115–19
"Music Maker" system 138–49
"Music Player" program 148–49
OPEN command 10
ORAZ ML instruction 70
output 18–22
"Paddle Graphics" program 87–90
"Page Flip" program 3–4
"Piano Player" program 120–27
"Pilot" program 266–69
"Place Your Bet" program 293–95
"Poker Challenge" program 205–8
precision, arithmetical 18–20
printer 46, 62–63
PRINT statement 24
PRINT USING statement 18
program listing conventions 329–31
programmable characters. *See* custom characters
Programming the 6502 72
Programming the VIC 31
quote mode 330
"Rainbow Border" program 136–37
raster beam register 9
"Rats!" program 220–32
READ statement 328
relative files 10–12
"Rescue Mission" program 303–5
rounding off numbers 19–20
"S.X. Demo" program 137
SAVE command 15–17
saving memory 23–25
"Scavenger Hunt" programs 306–11
screen 9, 58–59, 62–63
  and border color register 135
  dump, high-resolution 46
  management 3, 6, 81–83
  memory 5–8
  pages 3
  save 41–43
  split 9, 58–59
"Screen Print II" program 62–63
"Screen Saves" program 42–43
"Scroll Demo" program 30
scrolling 29–30
"Scroll Left" program 30
"Scroll Right" program 20–30
secondary address 15
SED ML instruction 70
serial printer 153
"Sevicog" program 264–65
6502 Assembly Language Programming 71
"Slot Machine" program 185–88
"Space Blockade" program 258–60
"Space Corridor" program 235–38
"Space Mission" program 279–85
"Special Effects" program 79–80
"Speed Demon" program 173–77
"Speed Stick Demo" program 37
"Speed Stick" program 36–37
split screen 9, 58–59
"Spring Man" program 318–20
"Sprite Demo" program 104
"Sprite Loader" program 101–4
sprites 95–104
  defining 96–100
  grid 99
  registers 97
"Strike Three" program 167–72
string variables 23–24
strings 158
STX ML instruction 70
"Super Dump" program 46–47
Super Expander cartridge 46, 135, 254, 264, 300
  checking for presence of 254
tape 13–17, 31–33, 153, 333
"Tape Catalog" program 66–69
tape header 31–32
tape programs
  cataloging 64–69
  copying to disk 44
"Tape Search" program 31–33
"Tape to Disk" program 44–45
text, mixing with graphics characters 9
text adventures 286–88
"Torpedo-8" program 261–63
"Trench Wars" program 321–24
typing in programs 327–31
unexpanded VIC 29, 31, 34, 57, 92, 95, 153, 158, 162, 167, 173, 178, 185, 199, 205, 215, 217, 235, 239, 241, 246, 249, 251, 258, 261, 270, 273, 279, 293, 296, 303, 306, 312, 315, 318, 321
USR function 81–82
variables 23–24
VERIFY command 16
Versatile Interface Adapter. *See* via
VIA 36
VIC chip 6, 62, 91
"VIC Draw McGraw" program 105–14
"VIC Graph" utility 91–94
"VIC Marquee" program 158–61
"VIC Set Demo" program 8
"VIC Set" program 8
VIC-20 Programmer's Reference Guide 71
Video Interface Chip. *See* VIC chip
WAIT statement 9
word processor 153–57
"Write-On" word processor 153–57

A VIC-20 Bargain

This huge collection of over 70 never-before-published articles and programs has something for every VIC-20 user. *COMPUTE!'s VIC-20 Collection* is a potpourri of tutorials, utilities, applications, and games collected over the past few years.

Do you like to play games? Then you'll enjoy maze games like "Heist" and "Meltdown" that challenge you to think quickly. If logic games are your specialty, you'll find "Logicolor" and "Maneuver" intriguing puzzlers. And of course we've included adventure and arcade-style games to amuse and entertain.

Do you need a simple word processor or a way to create sprites on your VIC? Maybe you just bought a disk drive and want a quick and easy way to transfer all your programs from tape to disk. You'll find these and much more in *COMPUTE!'s VIC-20 Collection*. Here's a sample of what's inside:

- "Message Board," an easy way to create a message that will scroll across your screen while another program is running
- "Music Mate," a music generator
- A disassmbler to help you turn numbers into readable code
- Nearly three dozen games, most of which run on an unexpanded VIC
- Techniques to create your own character set
- A simple way to have two BASIC programs running on your VIC at the same time
- Programs that let you copy the screen to a printer
- And more than 25 tutorials and utilities

All the programs in *COMPUTE!'s VIC-20 Collection* are ready to type in and use. And to make program entry as mistake-proof as possible, we've included a program that checks your typing. Many of the programs in this book are beginner-level programs. While they lack the sophistication of professional software, they'll provide hours of entertainment and tutorial opportunities for modification and enhancement.

ISBN 0-87455-007-6